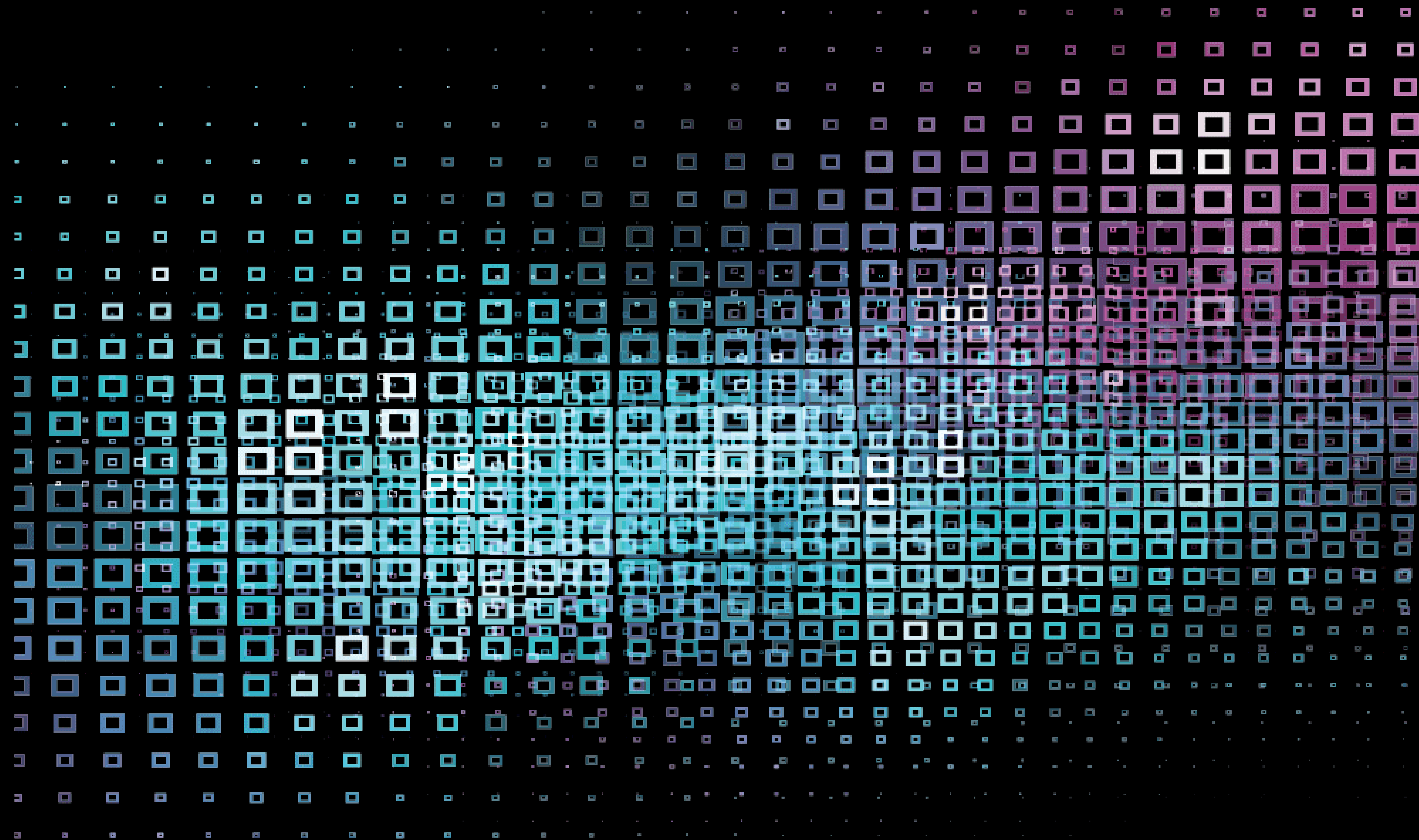
软件工程师开发大系

Java Web
开发实例大全（提高卷）

600个典型实例及源码分析，涵盖23个应用方向

✔工作应用**速查** ✔项目开发**参考** ✔学习实战**练习**

软件开发技术联盟◎编著



▶ **工作应用速查：** 本书实例全面、系统，涉及程序开发的各个方面，适合各级程序开发人员速查速用。

▶ **项目开发参考：** 程序员借助本书提供的实例源代码，可以快速搭建工程项目，提高开发效率。

▶ **学习实战练习：** 入门者的实战训练大全，不但可以激发学习兴趣，更可提高编程实战能力和编程思维水平。



清华大学出版社

软件工程师开发大系

Java Web 开发实例大全

（提高卷）

软件开发技术联盟 编著

清华大学出版社

北 京

内 容 简 介

《Java Web 开发实例大全（提高卷）》筛选、汇集了 Java Web 开发从基础知识到高级应用各个层面的大量实例及源代码，共有 600 个左右，每个实例及源代码按实例说明、关键技术、设计过程、详尽注释、秘笈心法的顺序进行了分析解读。全书分为 7 篇 23 章，包括流行组件应用、数据库应用、图表统计、Ajax 框架应用、流行框架、网站安全与架构模式、综合应用等。重点内容有操作 XML 文件、发送与接收邮件、数据库操作技术、SQL 语句应用技术、复杂查询技术、数据库高级应用、JFreeChart 绘图基础、基础图表技术、扩展图表技术、基于 Cewolf 组件的图表编程、Prototype 框架、jQuery 框架、Dojo 框架、Struts2 框架应用、Struts2 框架标签应用、Hibernate 框架基础、Hibernate 高级话题、Spring 框架基础、Spring 的 Web MVC 框架、网站性能优化与安全策略、设计模式与架构、网站设计与网页配色、Java Web 典型项目开发案例等。配书光盘附带了实例的源程序。

《Java Web 开发实例大全（提高卷）》既适合 Java Web 程序员参考和查阅，也适合 Java Web 初学者，如高校学生、软件开发培训学员及相关求职人员学习、练习、速查使用。

图书在版编目（CIP）数据

Java Web 开发实例大全．提高卷/软件开发技术联盟编著．—北京：清华大学出版社，2016
（软件工程师开发大系）
ISBN 978-7-302-38475-5

I. ①J…　II. ①软…　III. ①JAVA 语言-程序设计　IV. ①TP312

中国版本图书馆 CIP 数据核字（2014）第 260798 号

责任编辑：赵洛育
封面设计：李志伟
版式设计：郑　坤
责任校对：赵丽杰
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：203mm×260mm　　**印　　张：**61.25　　**字　　数：**2023 千字
（附光盘 1 张）
版　　次：2016 年 1 月第 1 版　　**印　　次：**2016 年 1 月第 1 次印刷
印　　数：1～3500
定　　价：128.00 元

产品编号：052256-01

前　言

Preface

特别说明：

《Java Web 开发实例大全》分为基础卷和提高卷（即本书）两册。本书的前身是《Java Web 开发实战 1200 例（第 II 卷）》。

编写目的

1. 方便程序员查阅

程序开发是一项艰辛的工作，挑灯夜战、加班加点是常有的事。在开发过程中，一个技术问题可能会占用几天甚至更长时间。如果有一本开发实例大全可供翻阅，从中找到相似的实例作参考，也许几分钟就可以解决问题。本书编写的主要目的就是方便程序员查阅、提高开发效率。

2. 通过分析大量源代码，达到快速学习之目的

本书提供了约 600 个开发实例及源代码，附有相应的注释、实例说明、关键技术、设计过程和秘笈心法，对实例中的源代码进行了比较透彻的解析。相信这种办法对激发学习情趣、提高学习效率极有帮助。

3. 通过阅读大量源代码，达到提高熟练度之目的

俗话说“熟能生巧”，读者只有通过阅读、分析大量源代码，并亲自动手去做，才能够深刻理解、运用自如，进而提高编程熟练度，适应工作之需要。

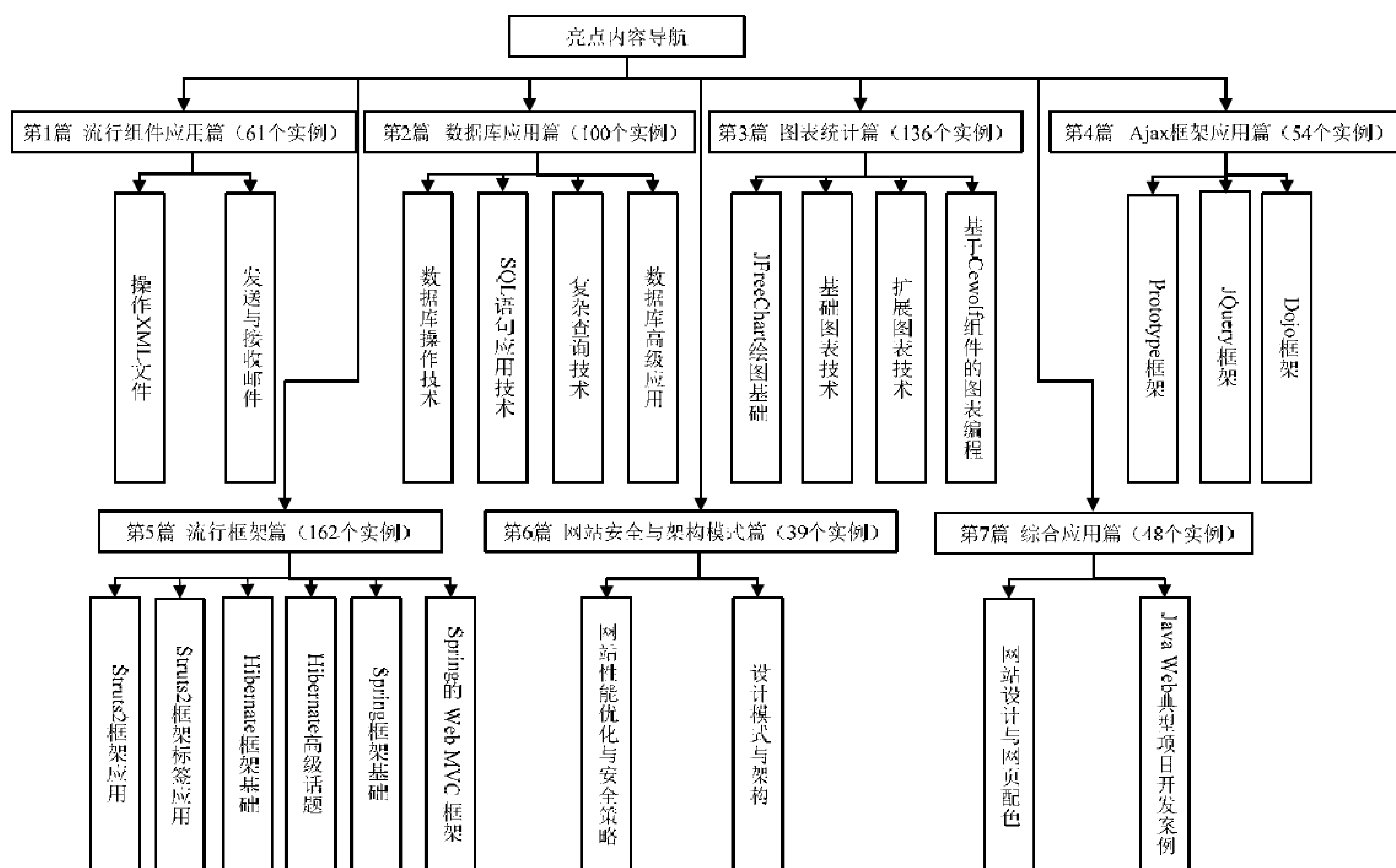
4. 实例源程序可以“拿来”就用，提高了效率

本书的很多实例，可以根据实际应用需求稍加改动，拿来就用，不必再去从头编写，从而节约时间，提高工作效率。

本书内容

本书精选了 600 个实例，涵盖了操作 XML 文件、发送与接收邮件、数据库操作技术、SQL 语句应用技术、复杂查询技术、数据库高级应用、JFreeChart 绘图基础、基础图表技术、扩展图表技术、基于 Cewolf 组件的图表编程、Prototype 框架、jQuery 框架、Dojo 框架、Struts2 框架应用、Struts2 框架标签应用、Hibernate 框架基础、Hibernate 高级话题、Spring 框架基础、Spring 的 Web MVC 框架、网站性能优化与安全策略、设计模式与架构、网站设计与网页配色、Java Web 典型项目开发案例等各方面的内容，每个知识点下还提供了针对性和实用性较强的经验技巧，帮助开发人员快速解决疑难问题。本书知识结构如下图所示。

本书在讲解实例时采用统一的编排样式，多数实例由“实例说明”“关键技术”“设计过程”“秘笈心法”4 部分构成。其中，“实例说明”部分采用图文结合的方式介绍实例的功能和运行效果；“关键技术”部分介绍了实例使用的重点、难点技术；“设计过程”部分讲解了实例的详细开发过程；“秘笈心法”部分给出了与实例相关的技巧和经验总结。



本书特点

1. 实例极为丰富

本书精选了 600 个实例，另外一册《Java Web 开发实例大全（基础卷）》也精选了约 600 个实例，这样，两册图书总计约 1200 个实例，可以说是目前市场上实例最多、知识点最全面、内容最丰富的软件开发类图书，涵盖了编程中各个方面的应用。

2. 程序解释详尽

本书提供的实例及源代码，附有相应的注释、实例说明、关键技术、设计过程和秘笈心法。分析解释详尽，便于快速学习。

3. 实践实战性强

本书的实例及源代码很多来自现实开发实践，光盘中给出了绝大多数实例的全部源代码，读者可以直接调用、研读、练习。

关于光盘

1. 实例学习注意事项

读者在按照本书学习、练习的过程中，可以从光盘中复制源代码，修改时注意去掉源码文件的只读属性。**有些实例需要使用相应的数据库或第三方资源，在使用前需要进行相应配置，具体步骤请参考书中或者光盘中的配置说明。**

2. 实例源代码

本书光盘提供了实例的源代码，位置在光盘中的“MR\章号\实例序号”文件夹下，例如，“MR\04\096”表示实例 096，位于第 4 章。**由于有些实例源代码较长，限于篇幅，图书中只给出了关键代码，完整代码放置在光盘中。**

读者对象

Java Web 程序员，Java Web 初学者，如高校大学生、求职人员、培训机构学员等。

本书服务

如果您使用本书的过程中遇到问题，可以通过如下方式与我们联系。

☑　服务 QQ：4006751066

☑　服务网站：http://www.mingribook.com

本书作者

本书由软件开发技术联盟组织编写，参与编写的程序员有赛奎春、王小科、王国辉、王占龙、高春艳、张鑫、杨丽、辛洪郁、周佳星、申小琦、张宝华、葛忠月、王雪、李贺、吕艳妃、王喜平、张领、杨贵发、李根福、刘志铭、宋禹蒙、刘丽艳、刘莉莉、王雨竹、刘红艳、隋光宇、郭鑫、崔佳音、张金辉、王敬洁、宋晶、刘佳、陈英、张磊、张世辉、高茹、陈威、张彦国、高飞、李严。在此一并致谢！

编　者

目　录

contents

第 1 篇　流行组件应用篇

第 1 章　操作 XML 文件 2
1.1　XML 基础操作 3
实例 001　CSS 格式化 XML 布局 3
实例 002　CSS 改变 XML 中鼠标指针形状 5
实例 003　CSS 在 XML 中添加背景图 7
实例 004　CSS 制作 XML 表格 8
实例 005　XML 中提取节点字符串值 10
实例 006　在 XML 内部定义 DTD 12
实例 007　在 XML 外部引用 DTD 14
实例 008　验证 XML 是否符合 DTD 的定义 15
实例 009　在 DTD 中声明元素 16
实例 010　在 DTD 中声明重复元素 17
实例 011　在 DTD 中声明选择性元素 19
实例 012　在 DTD 中使用 ENTITY 20
1.2　应用 XML Schema 21
实例 013　验证 XML 是否符合 Schema 的描述 21
实例 014　XSD 文档根元素的引用 24
实例 015　在 XSD 中设定元素的出现顺序 25
实例 016　在 XSD 中使用扩展数据类型 26
实例 017　在 XSD 中使用元素的条理化 29
实例 018　XSD 中的多属性打包 30
实例 019　XSD 中对元素的限定 32
实例 020　在 XSD 中使用取值范围的限定 34
实例 021　在 XSD 中声明元素属性 36
实例 022　在 XSD 中对字符进行限制 38
实例 023　在 XSD 中对数值进行限制 39
1.3　XML 解析 41
实例 024　使用 DOM 组件从文件中读取 XML 41
实例 025　使用 DOM 组件从数据流中读取 XML 42
实例 026　使用 JDOM 组件从文件中读取 XML 44
实例 027　使用 JDOM 组件读取 XML 45
实例 028　使用 SAX 组件从文件中读取 XML 48
实例 029　使用 SAX 组件从数据流中读取 XML 50
实例 030　使用 DOM 组件解析 XML 元素名称 51
实例 031　使用 DOM 组件解析 XML 元素名称和内容 53
实例 032　使用 SAX 组件解析 XML 元素名称 56
实例 033　使用 SAX 组件解析 XML 元素名称和内容 57
实例 034　使用 SAX 组件解析 XML 元素属性和属性值 61
实例 035　使用 DOM 组件解析 XML 元素属性和属性值 63
实例 036　使用 SAX 验证 DTD 65
实例 037　使用 dom4j 解析 XML 文件 67

第 2 章　发送与接收邮件 70
2.1　配置邮件服务器 71
实例 038　在 Windows Server 2003 系统下安装和配置邮件服务器 71
实例 039　配置开源邮件服务器 Apache James Server 73
实例 040　安装和配置 Magic Winmail 邮件服务器 74
2.2　应用 JavaMail 组件发送邮件 76
实例 041　发送普通格式的邮件 77
实例 042　发送 HTML 格式的邮件 79
实例 043　发送带附件的邮件 81
实例 044　群发普通邮件 83
实例 045　群发 HTML 格式的邮件 85
实例 046　群发带附件的邮件 86
实例 047　通过邮箱激活用户的注册 87
2.3　应用 JavaMail 组件接收邮件 90
实例 048　应用 POP3 协议接收未读邮件和已读邮件 90
实例 049　应用 POP3 协议接收带附件的邮件 95
实例 050　应用 IMAP 协议接收未读邮件和已读邮件 101

实例 051 应用 IMAP 协议接收带附件的邮件 104
2.4 应用 Apache commons-email 组件发送邮件 107
实例 052 发送普通格式的邮件 107
实例 053 发送带多个附件的邮件 109
实例 054 群发普通邮件 111
实例 055 群发 HTML 格式的邮件 112
实例 056 群发带附件的邮件 113
实例 057 通过邮箱激活用户的注册 114
2.5 应用 Spring 的 E-mail 抽象层发送邮件 ... 117
实例 058 发送普通文本邮件 117
实例 059 发送 HTML 格式的邮件 119
实例 060 发送带附件的邮件 122
实例 061 群发普通文本邮件 124

第 2 篇　数据库应用篇

第 3 章　数据库操作技术 128
3.1 建立 Connection 数据库连接 129
实例 062 建立 Access 数据库连接 129
实例 063 建立与 MySQL 数据库的连接 130
实例 064 建立与 SQL Server 2000 数据库的连接 131
实例 065 建立与 SQL Server 2005 数据库的连接 132
实例 066 建立与 Oracle 数据库的连接 133
实例 067 建立与 Java DB 数据库的连接 134
3.2 数据库与数据表 135
实例 068 列举 SQL Server 数据库中的数据表 135
实例 069 列举 MySQL 数据库中的数据表 136
实例 070 查看数据表结构 137
实例 071 动态维护投票数据库 138
实例 072 SQL Server 数据库的备份 141
实例 073 SQL Server 数据库的恢复 144
实例 074 MySQL 数据库的备份 147
实例 075 MySQL 数据库的恢复 149
3.3 数据库的添加、删除与更新操作 150
实例 076 将员工信息添加到员工表 150
实例 077 在添加数据时进行数据验证 151
实例 078 插入用户登录日志信息 152
实例 079 生成有规律的编号 153
实例 080 生成没有规律的编号 155
实例 081 在插入数据时过滤危险字符 156
实例 082 将用户选择的爱好信息以字符串形式保存到数据库 157
实例 083 实现跨数据库的表内容复制 158
实例 084 使用 UNION ALL 语句批量插入数据 158
实例 085 更新指定记录 159
实例 086 将数据表清空 161
第 4 章　SQL 语句应用技术 162
4.1 聚集函数与日期查询 163
实例 087 利用 SUM 函数实现数据汇总 163
实例 088 利用 AVG 函数实现计算平均值 164
实例 089 利用 MIN 函数求数据表中的最小数据 165
实例 090 利用 MAX 函数求数据表中的最大值 166
实例 091 利用 COUNT 函数求销售额大于某值的图书种类 167
实例 092 查询与张静同一天入司的员工信息 168
实例 093 使用 IN 谓词查询某几个时间的数据 169
实例 094 对数据进行降序排序查询 171
实例 095 数据的多条件排序查询 172
实例 096 对统计结果进行排序 173
实例 097 查询 SQL Server 数据表中的前 3 条数据 176
实例 098 查询 SQL Server 数据表中的后 3 条数据 177
实例 099 查询 MySQL 数据表中的前 3 条数据 178
实例 100 查询 MySQL 数据表中的后 3 条数据 179
4.2 排序与分组函数的应用 180
实例 101 按照字母顺序对留学生表进行排序 180
实例 102 按姓氏笔画排序 182
实例 103 将汉字按音序排序 183
实例 104 按列的编号排序 184
实例 105 从表中随机返回记录 185
实例 106 使用 GROUP BY 子句实现数据的分组统计 186
实例 107 利用 GROUP BY 子句实现多表分组统计 187
4.3 比较大小与逻辑应用 189

实例 108　在查询结果中不显示重复记录 189
实例 109　使用 NOT 查询不满足条件的记录 190
实例 110　使用 BETWEEN 进行区间查询 192
实例 111　使用关系运算符查询某一时间段的数据 193
实例 112　计算两个日期间的月份数 194

第 5 章　复杂查询技术 196
5.1　使用子查询 197
实例 113　将子查询作为表达式 197
实例 114　用子查询作为派生表 198
实例 115　通过子查询关联数据 199
实例 116　使用 IN 谓词限定查询范围 200
实例 117　使用 NOT IN 子查询实现差集运算 202
实例 118　使用 NOT IN 子查询实现反向查询 203
实例 119　实现笛卡儿乘积查询 204
实例 120　比较运算符引入子查询 205
实例 121　在查询中使用聚合函数 206
实例 122　在删除数据时使用子查询 207
5.2　多表连接查询 208
实例 123　使用 UNION 运算符使学生档案归档 208
实例 124　内连接查询指定课程的教师信息 209
实例 125　左外连接查询员工信息 210
实例 126　右外连接查询员工信息 212
实例 127　多表外连接查询 213
实例 128　完全连接查询 214
5.3　嵌套查询 215
实例 129　查询平均成绩在 85 分以上的学生信息 215
实例 130　多表统计本科学历部门经理的月收入情况 216
实例 131　在嵌套中使用 EXISTS 关键字 217
实例 132　动态指定查询条件 218
5.4　常见谓词的使用 219
实例 133　应用 PATINDEX 谓词进行模糊查询 219
实例 134　在查询中使用四舍五入谓词 ROUND 220
实例 135　查询比质量部所有员工工资都高的员工信息 221
实例 136　查询工资高于质量部任意一名员工的员工信息 222
实例 137　应用 UNION 谓词消除重复的行 223
实例 138　应用 UNION ALL 谓词保留重复行 224
实例 139　查询各商品销售额所占的百分比 225

第 6 章　数据库高级应用 227
6.1　在 Java Web 程序中调用存储过程 228
实例 140　调用存储过程实现用户身份的验证 228
实例 141　调用存储过程添加数据 229
实例 142　调用加密存储过程 230
实例 143　获取数据库中所有存储过程 231
实例 144　修改存储过程 233
实例 145　删除存储过程 234
6.2　使用触发器 235
实例 146　应用触发器添加日志信息 235
实例 147　应用触发器级联删除数据 237
实例 148　调用 UPDATE 触发器修改数据 238
实例 149　获取数据库中所有触发器名称 240
实例 150　创建带有触发条件的触发器 240
6.3　使用批处理 242
实例 151　使用批处理删除数据 242
实例 152　批量提高员工工资 245
实例 153　将教师表中数据全部添加到选课表 246
实例 154　在批处理中使用事务 248
6.4　使用视图 249
实例 155　通过 Java Web 程序创建视图 249
实例 156　应用视图查询数据 252
实例 157　使用视图计算数据 253
实例 158　使用视图格式化检索出来的数据 254
实例 159　获取数据库中的全部用户视图 255
实例 160　修改视图 256
实例 161　删除视图 257

第 3 篇　图表统计篇

第 7 章　JFreeChart 绘图基础 262
7.1　图表的基础 263
实例 162　基本饼图 263
实例 163　显示图例 265

实例 164　工具栏提示......266
实例 165　乱码问题......267
实例 166　显示数值......269
实例 167　抗锯齿设置......270
7.2　设置图表的背景......272
实例 168　设置背景图......272
实例 169　设置背景图片透明度......273
实例 170　设置背景色......275
7.3　处理图表的边框......276
实例 171　隐藏图表边框......276
实例 172　图表边框颜色和笔触......277
7.4　修改图表的图例......279
实例 173　设置图例背景色......279
实例 174　设置图例边框......280
实例 175　设置图例边缘间距......281
实例 176　设置图例字体颜色......283
实例 177　设置图例位置......284
第 8 章　基础图表技术......286
8.1　普通饼图......287
实例 178　分离饼图......287
实例 179　椭圆形饼图......288
实例 180　饼图的阴影......289
实例 181　加粗饼图分类边框......290
实例 182　设置饼图颜色......291
实例 183　饼图旋转角度和顺序......293
实例 184　隐藏分类标签连接线......294
8.2　3D 饼图......296
实例 185　创建 3D 饼图......296
实例 186　3D 饼图透明度......297
实例 187　3D 饼图的 Z 轴......298
8.3　多饼图......299
实例 188　实现多饼图......299
实例 189　多饼图乱码......301
实例 190　3D 多饼图......303
8.4　基本柱形图......304
实例 191　简单柱形图......304
实例 192　柱形图角度......306
实例 193　柱形图负值......308
8.5　X 坐标轴......309
实例 194　X 轴字体......309
实例 195　X 轴标签字体......310
实例 196　X 轴标签角度......312
实例 197　X 轴尺度线颜色......314
实例 198　隐藏 X 轴尺度线......315
实例 199　X 轴尺度线笔触......317
实例 200　X 轴尺度标签......319
实例 201　X 轴分类的间距......320
实例 202　X 轴分类与原点的间距......322
实例 203　X 轴的显示位置......323
8.6　Y 坐标轴......325
实例 204　Y 轴字体......325
实例 205　Y 轴标签字体......326
实例 206　Y 轴显示情况......328
实例 207　Y 轴尺度线颜色和笔触......329
实例 208　隐藏 Y 轴尺度线......331
实例 209　Y 轴尺度标签角度......333
实例 210　Y 轴起始值......334
实例 211　Y 轴箭头......335
实例 212　隐藏 Y 轴主要刻度线......337
实例 213　Y 轴主要刻度线长度......338
实例 214　设置 Y 轴最大值......340
实例 215　设置 Y 轴数据范围......341
实例 216　Y 轴的显示位置......343
8.7　高级柱形图......344
实例 217　设置网格竖线......344
实例 218　设置网格竖线颜色......346
实例 219　设置柱形图文本注解......347
实例 220　设置柱形图文本注解字体......349
实例 221　设置柱形图文本注解锚点......351
实例 222　设置柱形图文本注解的类别锚点......352
实例 223　设置柱形图文本注解的旋转锚点......354
实例 224　设置柱形图线条注解......356
实例 225　绘制柱形效果......357
实例 226　柱形图阴影......359
实例 227　柱形图阴影偏移......360
实例 228　设置柱形的颜色......362
实例 229　绘制 3D 柱形图......363
实例 230　标记柱形图区间......365
实例 231　多系列柱形图......367
实例 232　多系列 3D 柱形图......369

第 9 章　扩展图表技术 371
9.1　区域图 372
实例 233　基本区域图 372
实例 234　显示多分类区域图 373
实例 235　设置区域图透明度 375
实例 236　添加说明文字 377
实例 237　设置说明文字位置 379
实例 238　设置区域图 X 轴显示位置 381
实例 239　设置区域图 X 轴标签角度 383
实例 240　设置区域图 X 轴尺度标签角度 385
实例 241　设置区域颜色 386
9.2　折线图 388
实例 242　创建基本折线图 388
实例 243　创建多条折线图 390
实例 244　创建水平折线图 392
实例 245　隐藏折线图中指定系列的折线 394
实例 246　加粗折线 395
实例 247　显示折线节点 397
实例 248　生成节点图 399
实例 249　绘制虚线折线图 401
实例 250　设置折线颜色 403
实例 251　3D 折线图 404
实例 252　XY 折线图 407
实例 253　排序折线图 409
9.3　时序图 410
实例 254　基本时序图 410
实例 255　设置时间显示格式 412
实例 256　添加双时间轴 414
实例 257　设置双时间轴位置 415
实例 258　动态显示十字标记 417
实例 259　添加 Y 轴标记 419
实例 260　添加 X 轴标记 421
实例 261　设置刻度单位 422
实例 262　设置时间轴范围 424
9.4　联合分类图 425
实例 263　生成线形图与柱形图 425
实例 264　设置图表高度 427
实例 265　设置图表位置 429
9.5　图表的综合应用 431
实例 266　利用饼图分析不同编程语言的市场占有率 431
实例 267　利用柱形图显示某 Ajax 网站不同框架的年下载量 433
实例 268　利用折线图分析不同城市气温变化情况 434
实例 269　利用区域图分析不同学生的成绩变化 435
实例 270　利用时序图分析股票价格走势 436
实例 271　利用时序图分析 2009 年国际原油价格走势 438
实例 272　利用组合图表分析学生零用钱收支情况 439

第 10 章　基于 Cewolf 组件的图表编程 442
10.1　生成基于 DefaultCategoryDataset 数据集的图表 443
实例 273　生成水平直方图 443
实例 274　生成水平堆栈图 445
实例 275　绘制 3D 垂直直方图 447
实例 276　生成垂直堆栈图 448
实例 277　生成区域图 449
10.2　绘制饼状图表 450
实例 278　生成普通饼图 450
实例 279　生成 3D 饼图 452
10.3　绘制基于 XYDataset 数据集的图表 453
实例 280　生成线段图（折线图） 453
实例 281　生成区域图 455
实例 282　生成散列图 456
实例 283　生成时序图 457
实例 284　生成直方图 459
10.4　绘制基于 OHLCDataset 数据集的图表 460
实例 285　生成 K 线图 460
实例 286　生成高低图（HighLow） 462
10.5　生成组合图表 463
实例 287　生成水平组合图表 463
实例 288　生成垂直组合图表 465
10.6　绘制其他类型的图表 466
实例 289　生成甘特图 466
实例 290　生成罗盘图 468
实例 291　生成速度图 469
10.7　综合图表的应用 471

实例 292　利用柱形图对比不同城市的房价 471
实例 293　利用饼图显示投票结果 472
实例 294　利用折线图分析某城市蔬菜价格走势 474
实例 295　利用区域图对比分析员工业绩 475
实例 296　利用时序图分析商品月销售收益 477
实例 297　利用组合图表分析国际原油价格走势 479

第 4 篇　Ajax 框架应用篇

第 11 章　Prototype 框架 482
11.1　使用 Prototype 基本函数 483
实例 298　使用$()函数获取页面元素 483
实例 299　使用$A()函数实现将参数转换为数组 484
实例 300　使用$F()函数获取表单输入控件的值 485
实例 301　使用 Try.these()函数获取返回值 486
11.2　Prototype 自定义对象和类 487
实例 302　在 HTML 元素中增加 CSS 样式 487
实例 303　利用 Enumerable 对象在页面中显示数组元素 488
实例 304　使用 Field 对象操作表单域 490
实例 305　通过 Form 对象使表单元素失效 491
实例 306　使用 Form.Element 对象返回特定表单域的值 493
11.3　对 Ajax 的支持 494
实例 307　Ajax.Request 对象发送请求 494
实例 308　注册全局的事件处理器 495
实例 309　定时刷新时间 497
第 12 章　jQuery 框架 499
12.1　DOM 技术 500
实例 310　获取文本框中的文本 500
实例 311　利用 jQuery 实现查找节点 502
实例 312　动态为表格追加样式 503
实例 313　动态为表格移除样式 504
实例 314　实现表格的样式切换 506
12.2　表单处理 507
实例 315　实现表单文本域的放大和缩小 507
实例 316　实现复选框的全选与反选 508
实例 317　列表框的综合应用 509
实例 318　实现表单验证 511
实例 319　密码强度检测 512
实例 320　文本框提示标签 513
12.3　操作表格 514
实例 321　表格隔行变色 514
实例 322　通过单选按钮控制表格的行高亮显示 515
实例 323　通过复选框控制表格的行高亮显示 517
实例 324　表格的展开与关闭 518
实例 325　利用文本框的值实现对表格内容的筛选 519
12.4　其他特效 520
实例 326　制作网页选项卡 520
实例 327　日期拾取器 521
实例 328　网页软键盘 522
实例 329　图片幻灯片 523
实例 330　颜色拾取器 524
实例 331　广告轮显 525
实例 332　图片放大镜 527
实例 333　文本编辑器 528
实例 334　右键菜单 529
实例 335　结合 jQuery 实现在线裁剪 531
12.5　对 Ajax 的支持 534
实例 336　检测用户名是否被占用 534
实例 337　验证用户登录 536
实例 338　基于 jQuery 的 Ajax 聊天室 538
第 13 章　Dojo 框架 540
13.1　Dojo 的常用 Widget 541
实例 339　实现网页按钮 541
实例 340　实现网页对话框 542
实例 341　实现日历功能 544
实例 342　实现网页的多页面 545
13.2　Dojo 的基本应用 546
实例 343　鼠标单击事件处理 546
实例 344　访问被监听方法的参数 548
实例 345　页面 HTML 元素的任意移动 549
实例 346　页面元素的相对移动 550
实例 347　带手柄的移动 552
13.3　Dojo 对 Ajax 的支持 553

实例 348　基本请求的发送 553
实例 349　请求队列的发送 555
实例 350　对象的字符串化 556
实例 351　表单请求发送 558

第 5 篇　流行框架篇

第 14 章　Struts2 框架应用 562
14.1　Struts2 的基本配置与零配置 563
实例 352　成绩统计器 563
实例 353　成绩排序 564
实例 354　用户的直接登录 566
实例 355　实现用户的中间退出 567
14.2　Struts2 数据校验与拦截器 568
实例 356　日期转换器 568
实例 357　实现空表单信息的提示 569
实例 358　计时拦截器 571
实例 359　等待拦截器 571
实例 360　权限验证拦截器 572
14.3　文件上传与下载 574
实例 361　单文件的上传 574
实例 362　上传错误信息的提示 575
实例 363　特定文件格式的上传 576
实例 364　限定上传文件的大小 577
实例 365　多文件的上传 577
实例 366　文件下载 579
14.4　Struts2 对 Ajax 的支持 580
实例 367　调试信息的输出 580
实例 368　数据校验错误信息的输出 581
实例 369　Action 中错误信息的输出 582
实例 370　显示 Action 的信息 582
实例 371　显示新闻列表 583
实例 372　页面的自动刷新 584
实例 373　访问注册页面出错 585
实例 374　无刷新实现登录 586
实例 375　无刷新实现注销 587
实例 376　实现标签页 588
实例 377　调试信息的输出 589
实例 378　数据的树状输出 590
实例 379　文件的树状显示 591
实例 380　动态加载数据 592
第 15 章　Struts2 框架标签应用 595
15.1　OGNL 语言 596
实例 381　访问 OGNL 上下文 596
实例 382　访问 ActionContext 资源 597
实例 383　用“#”过滤筛选集合 598
实例 384　用“#”构造 Map 600
实例 385　获取 Request 的 account 属性 600
实例 386　在资源文件中引用 OGNL 601
实例 387　在 struts.xml 中引用 OGNL 602
15.2　控制标签 603
实例 388　判断用户是否存在 603
实例 389　用户不存在的提示 604
实例 390　简单的计算器 605
实例 391　多集合的连接 606
实例 392　字符串的分割 607
实例 393　集合的混合合并 608
实例 394　筛选集合元素 609
15.3　数据标签 610
实例 395　Action 页面的引入 610
实例 396　JavaBean 的引用 611
实例 397　页面日期的输出 613
实例 398　页面日期的格式化输出 614
实例 399　计算日期的时间差 614
实例 400　声明资源的国际化 616
实例 401　JSP 页面的引入 617
实例 402　页面间数据的传递 618
实例 403　页面数据的设定 619
实例 404　变量值的页面输出 620
15.4　表单标签 621
实例 405　表单的输出 621
实例 406　用户名的填写 622
实例 407　简单的用户登录页面 623
实例 408　本地文件的浏览 624
实例 409　数据的默认选择 624

实例 410　页面中单选按钮的实现 625
实例 411　实现表单的提交 626
实例 412　实现下拉列表框 627
实例 413　具有自动完成功能的下拉列表框 628
实例 414　使用动态数据的下拉列表框 629
实例 415　复选框的实现 630
实例 416　实现可填写的复合框 630
实例 417　日期选择器 631
实例 418　联动选择框 632
实例 419　多级数据选择框 633

第 16 章　Hibernate 框架基础 634

16.1　操作实体对象 635

实例 420　将实体对象保存到数据库 635
实例 421　更新实体对象 638
实例 422　删除数据 640
实例 423　批量添加数据 642
实例 424　采用一对一关联添加数据 645
实例 425　采用一对多关联添加数据 647

16.2　HQL 与 QBC 检索方式 649

实例 426　分组统计 649
实例 427　利用统计函数 SUM 求销售总额 650
实例 428　利用统计函数 AVG 求某班学生的平均成绩 652
实例 429　利用统计函数 COUNT 统计当前注册用户人数 654
实例 430　利用 HQL 查询图书表中的所有数据 655
实例 431　利用 HQL 查询满足指定条件的数据 656
实例 432　HQL 绑定参数查询 658
实例 433　只返回一个检索对象 660
实例 434　限制返回结果的范围 661
实例 435　分页查询数据 663
实例 436　利用 QBC 检索字段为空的记录 665
实例 437　利用 QBC 检索不满足指定条件的记录 666
实例 438　QBC 忽略大小写查询 668
实例 439　利用 QBC 查询满足指定范围的所有记录 669
实例 440　利用 HQL 实现模糊查询 671
实例 441　利用 QBC 实现模糊查询 672
实例 442　HQL 在查询中使用统计函数 673
实例 443　利用 HQL 实现投影查询 674
实例 444　QBC 实现将查询结果排序 675
实例 445　HQL 内连接查询商品信息 676

第 17 章　Hibernate 高级话题 678

17.1　关联映射 679

实例 446　关联映射实现级联保存与更新 679
实例 447　建立商品表与商品类型表的双向关联 680
实例 448　实现商品表的自关联 682
实例 449　在持久化类方法中加入程序代码 683
实例 450　主键关联映射 684
实例 451　外键关联映射 686
实例 452　多对多单向关联映射学生表与科目表 688
实例 453　多对多双向关联映射学生表与科目表 689

17.2　Hibernate 检索策略 691

实例 454　一对多的立即检索策略 691
实例 455　多对一的立即检索策略 692
实例 456　一对多的延迟检索策略 693
实例 457　迫切左外连接查询 695

17.3　Hibernate 集合映射与事务应用 696

实例 458　通过映射 Set 集合实现添加数据 696
实例 459　通过映射 List 集合实现添加数据 698
实例 460　通过映射 Map 集合实现添加数据 699
实例 461　事务回滚的应用 700
实例 462　配置持久化类实现乐观锁的使用 701

第 18 章　Spring 框架基础 704

18.1　Spring 的依赖注入 705

实例 463　应用 Setter 注入法实现 Bean 的注入 705
实例 464　应用构造器注入法实现 Bean 的注入 706
实例 465　应用@Autowired 注解实现 Bean 的注入 708
实例 466　应用@Resource 注解实现 Bean 的注入 710
实例 467　零配置实现 Bean 的注入 711
实例 468　为 JavaBean 的集合对象注入属性值 713
实例 469　使用<prop>标签为 Java 持久属性集注入值 715
实例 470　按照 Bean 的名称自动装配 User 716
实例 471　按照 Bean 的类型自动装配 User 717
实例 472　配置 Bean 的延迟初始化 717
实例 473　通过<beans>设置统一的延迟初始化行为 718

实例 474　自定义 MyDateEditor 编辑器实现类型转换 719
实例 475　验证用户登录 720
18.2　Spring 的事务管理 722
实例 476　应用编程式事务管理向用户信息表插入数据 722
实例 477　应用编程式事务管理向学生信息表插入数据 725
18.3　Spring 的面向切面编程 726
实例 478　利用 Spring AOP 使日志输出与方法分离 726
实例 479　Spring AOP 实现用户注册 728
18.4　Spring 的持久化 730
实例 480　在 Spring 中利用 DAO 模式添加数据 730
实例 481　利用 JdbcTemplate 向员工信息表添加数据 732
实例 482　利用 JdbcTemplate 查询员工信息表 734
实例 483　利用 JdbcTemplate 更新指定员工信息 735
实例 484　使用 JdbcTemplate 调用存储过程查询商品 738
实例 485　使用 SimpleJdbcTemplate 添加图书信息 740
实例 486　使用 SimpleJdbcTemplate 查询指定图书信息 741
实例 487　在 Spring 中配置 DBCP 数据库连接池 743
实例 488　在 Spring 中使用占位符配置数据源 744
实例 489　使用 destroy-method 处理数据源 746
实例 490　Spring 分页显示图书信息 747
实例 491　整合 Spring 和 Hibernate 添加员工信息 749
实例 492　整合 Spring 和 Hibernate 批量添加用户信息 751
18.5　在 Spring 中生成非 HTML 输出 753
实例 493　利用 Spring 将学生信息导出到 Excel 工作表 753
实例 494　利用 Spring 将图书信息导出到 PDF 文件 756
18.6　Spring 文件上传与国际化 757
实例 495　利用 Spring 实现文件的上传 757
实例 496　利用 Spring 实现用户登录页面的国际化 760
第 19 章　Spring 的 Web MVC 框架 761
19.1　Spring 的控制器 762
实例 497　使用简单控制器获取表单数据 762
实例 498　参数映射控制器映射 JSP 页面 764
实例 499　文件名映射控制器映射 JSP 页面 765
实例 500　命令控制器获取 URL 中的参数查询信息 767
实例 501　利用表单控制器向图书信息表中添加数据 769
实例 502　利用表单控制器验证用户登录 772
实例 503　利用多动作控制器跳转到不同页面 774
实例 504　利用向导控制器实现用户注册 775
实例 505　利用多动作控制器操作员工信息表的数据 778
19.2　在线通讯录 780
实例 506　添加新联系人 780
实例 507　修改联系人信息 783
实例 508　删除联系人 785
实例 509　查询通讯录中的信息 786
19.3　图书信息管理 787
实例 510　添加图书信息 787
实例 511　修改图书信息 789
实例 512　删除图书信息 790
实例 513　查询图书信息 791

第 6 篇　网站安全与架构模式篇

第 20 章　网站性能优化与安全策略 794
20.1　文件保护 795
实例 514　防止用户直接输入地址访问 JSP 文件 795
实例 515　防止页面重复提交 797
实例 516　对查询字符串进行 URL 编码 800
实例 517　过滤非法字符 801
实例 518　禁止用户输入敏感字符 803
20.2　漏洞防护与数据加密 804

实例 519 文件上传漏洞 804
实例 520 防止资源被盗链下载 808
实例 521 对登录密码进行加密 809
实例 522 字符串加密 812
实例 523 MD5 加密注册用户名和密码 814
20.3 获取客户端信息 816
实例 524 确定对方的 IP 地址 816
实例 525 获取客户端 TCP/IP 端口的方法 817
实例 526 确定对方的浏览器信息 819
实例 527 确定对方浏览器可接收信息的类型 819
第 21 章 设计模式与架构 821
21.1 接口型模式 822
实例 528 适配器模式 822
实例 529 外观模式 823
实例 530 组合模式 826
实例 531 桥接模式 828
21.2 责任型模式 831
实例 532 单例模式 831
实例 533 建造者模式 833
实例 534 中介者模式 836
实例 535 责任链模式 838
实例 536 享元模式 841
实例 537 代理模式 843
21.3 构造型模式 844
实例 538 装饰模式 844
实例 539 工厂方法模式 846
实例 540 抽象工厂模式 849
实例 541 原型模式 851
实例 542 备忘录模式 853
21.4 行为型模式 855
实例 543 命令模式 855
实例 544 解释器模式 857
实例 545 迭代器模式 859
实例 546 观察者模式 861
实例 547 状态模式 864
实例 548 策略模式 866
实例 549 模板方法模式 868
实例 550 访问者模式 870
21.5 网站开发架构模式 872
实例 551 MVC 框架在联系人管理网站中的应用 872
实例 552 应用 MVC 架构开发简单计算器 876

第 7 篇 综合应用篇

第 22 章 网站设计与网页配色 882
22.1 企业网站 883
实例 553 汽车销售网 883
实例 554 医药连锁网 884
实例 555 硬件产品网 885
实例 556 软件产品网 886
实例 557 物流网 887
实例 558 宾馆酒店网 888
22.2 电子商务类 889
实例 559 B2C 电子商务网 889
实例 560 B2B 电子商务网 890
22.3 搜索引擎类 891
实例 561 站内搜索引擎 891
实例 562 互联网搜索引擎 892
22.4 生活资讯类 893
实例 563 家居销售网 894
实例 564 房地产信息网 895
22.5 娱乐类网站 896
实例 565 音乐网 896
实例 566 电影网 897
实例 567 游戏门户网 899
22.6 供求信息类 900
实例 568 人才供求网 900
实例 569 二手商品供求网 901
22.7 其他应用 902
实例 570 个人主页 902
实例 571 美食网 904
实例 572 博客网站 905
第 23 章 Java Web 典型项目开发案例 906
23.1 Ajax 聊天室 907

实例 573　实时获取并显示在线人员列表 907
实例 574　实现用户发言 909
实例 575　实时显示聊天内容 912
实例 576　安全退出聊天室 915
23.2　博客网核心模块开发 917
实例 577　注册自己的博客 917
实例 578　根据域名访问博客 919
实例 579　推荐博客设置 920
实例 580　文章浏览操作 922
23.3　在线投票统计功能 924
实例 581　实现投票功能 924
实例 582　实现柱形图统计功能 925
实例 583　实现饼图统计功能 928
实例 584　双击鼠标展开图片 930
23.4　B2C 电子商务网站 931
实例 585　添加商品到购物车 931
实例 586　查看购物车 933
实例 587　修改商品数量 935
实例 588　生成订单 936
23.5　在线音乐 939
实例 589　试听歌曲并同步显示歌词 939
实例 590　添加歌曲 941
实例 591　以顺序和随机方式进行歌曲连播 942
23.6　校内数码相册 946
实例 592　以幻灯片方式播放数码相片 946
实例 593　创建相册分类并上传相片 947
实例 594　浏览和管理上传相片 949
实例 595　数码相册分类管理 952
23.7　仿百度知道之明日知道 955
实例 596　在线提问 955
实例 597　问题回复 957
实例 598　修改问题 958
实例 599　关闭提出的问题 959
实例 600　搜索问题 960

第1篇

流行组件应用篇

⏭ 第 1 章　操作 XML 文件

⏭ 第 2 章　发送与接收邮件

第 1 章

操作 XML 文件

- XML 基础操作
- 应用 XML Schema
- XML 解析

1.1　XML 基础操作

XML（eXtensible Markup Language，可扩展标记语言）是由 W3C 定义的一种语言，推出这种语言的主要目的是使 Internet 上的数据交互更加方便，文件内容更加清晰易懂。下面通过相关实例来更好地理解 XML 的一些基础操作。

实例 001	CSS 格式化 XML 布局 光盘位置：光盘\MR\01\001	初级 实用指数：★★★★

实例说明

XML 最大的特点就是可以很好地揭示数据的本身含义，因此使用 XML 文档来描述、存储和共享数据是具有很多优点的，但是 XML 文档中却并不包含数据的显示格式信息，这样就会造成数据显示时视觉上的混乱。为了解决这一问题，就需要使用特定的样式表语言对数据的显示加以描述。本实例使用 W3C 提供的一种样式语言 CSS（Cascading Style Sheets，层叠样式表，是目前格式化 XML 文档内容的方法之一）来格式化 XML 文档。本实例的运行结果如图 1.1 所示。

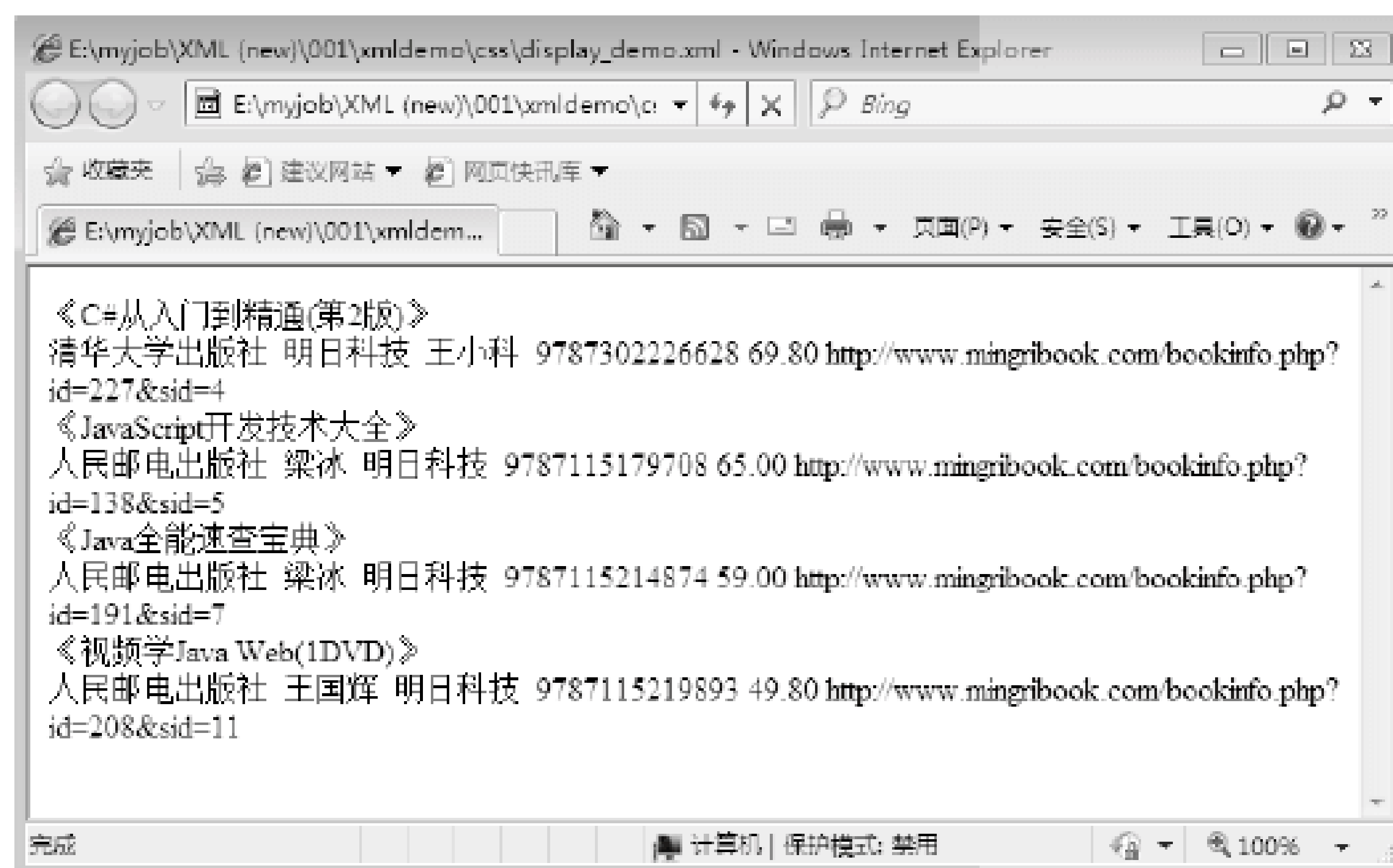


图 1.1　CSS 格式化 XML 文档

本实例主要应用 CSS 样式的字体属性、颜色和背景属性以及边框属性来实现。下面分别介绍这几种属性的用法。

（1）字体属性

字体属性包括字体的颜色、字体大小、字体的风格等。常用的字体属性如表 1.1 所示。

表 1.1　字体属性及说明

字 体 属 性	说　明
font	设置或者检索对象中文特性的复合属性
font-family	用于指定字体名称，可以指定一个字体名，也可以用“,”分隔指定多个字体名
font-size	设置字体的字号
font-variant	设置英文大小写转换
font-weight	设置字体的粗细
font-style	用于指定是否对字体应用斜体风格

（2）颜色和背景属性

CSS 中的颜色属性用于设置页面元素的颜色，背景属性用于设置背景色或背景图像，具体属性及说明如表 1.2 所示。

表 1.2　颜色和背景属性及说明

颜色和背景属性	说　明
color	设置页面的前景色
background-color	设置背景色
background-image	设置背景图像

续表

颜色和背景属性	说　明
background-repeat	设置背景图像的排列方式。有 4 种，分别是 repeat（在水平和垂直方向上都是以瓷砖形式反复显示）、repeat-x（只在水平方向重复显示）、repeat-y（只在垂直方向重复显示）、no-repeat（不重复，只显示一个）
background-attachment	设置背景图像是否固定，其值包括 fixed（固定背景图像）和 scroll（背景图像和其他内容一起滚动）
background-position	设置背景图像的显示位置
background	综合设置背景图像的属性

（3）边框属性

边框属性用于设置元素的边框宽度、样式和颜色，具体说明如表 1.3 所示。

表 1.3　边框属性及说明

边 框 属 性	说　明	边 框 属 性	说　明
border	边框复合属性	border-right-color	右边框颜色
border-top	上边框	border-bottom-color	下边框颜色
border-left	左边框	border-top-style	上边框样式
border-right	右边框	border-left-style	左边框样式
border-bottom	下边框	border-right-style	右边框样式
border-color	边框颜色	border-top-width	上边框粗度
border-style	边框样式	border-left-width	左边框粗度
border-width	边框粗度	border-right-width	右边框粗度
border-top-color	上边框颜色	border-bottom-width	下边框粗度
border-left-color	左边框颜色		

边框样式的属性值及说明如表 1.4 所示。

表 1.4　边框样式的属性值及说明

边框样式属性值	说　明	边框样式属性值	说　明
none	无边框	double	边框是双实线
hidder	隐藏边框，IE 不支持	groove	边框带有立体感的沟槽
dotted	边框由点组成	ridge	边框呈脊形
dashed	边框由短线组成	inset	边框内嵌一个立体边框
solid	边框是实线	outset	边框外嵌一个立体边框

设计过程

（1）编写 CSS 样式表文件 style.css，在该文件中指定 XML 文档中各元素的显示样式。代码如下：

```
<--指定各元素显示样式-->
book{
border:1px solid;
width:100%;
font-size:12px;
}
ISBN{
display: none;
}
bookname{
width: 200px;
}
author{
font-style: italic;
width: 200px;
```

```
text-align:center;
}
publishing{
width: 100px;
}
```

（2）编写 index.xml 文件，在该文件中创建一个 books 根元素，该元素由多个 book 元素组成。代码如下：

```
<?xml version="1.0" encoding="gb2312" ?>
<!--声明 XML 文档版本与字符编码方式-->
<books>
<--创建根元素 books-->
<book>
<--创建子元素 book-->
      <ISBN>7-302-21033-7</ISBN>
      <bookname>Java Web 开发实战宝典</bookname>
      <author>王国辉</author>
      <publishing>清华大学出版社</publishing>
</book>
<book>
<--创建子元素 book-->
      <ISBN>7-115-14689-6</ISBN>
      <bookname>JSP 数据库系统开发案例精选</bookname>
      <author>王国辉、王易</author>
      <publishing>人民邮电出版社</publishing>
</book>
</books>
```

（3）在 index.xml 文件的第 2 行加入如下引用 CSS 样式表文件的代码。

```
<?xml-stylesheet href="style.css" type="text/css"?>
<--引入 CSS 样式表-->
```

秘笈心法

心法领悟 001：XML 样式文件的导入。

一个 XML 文档可能有多种风格样式，而且文档中的不同元素风格也可能不同。为了实现不同风格样式的切换，可以定义多个不同风格样式的 CSS 样式文件，在文档中需要使用时，直接通过<href>导入即可。

实例 002　CSS 改变 XML 中鼠标指针形状

光盘位置：光盘\MR\01\002

初级

实用指数：★★★☆

如果希望控制鼠标指针在运动到文字的显示区域时的形状，可以使用 CSS 中的相关属性进行设置。本实例实现指定鼠标运动到文字的显示区域时变成手的形状，运行结果如图 1.2 所示。

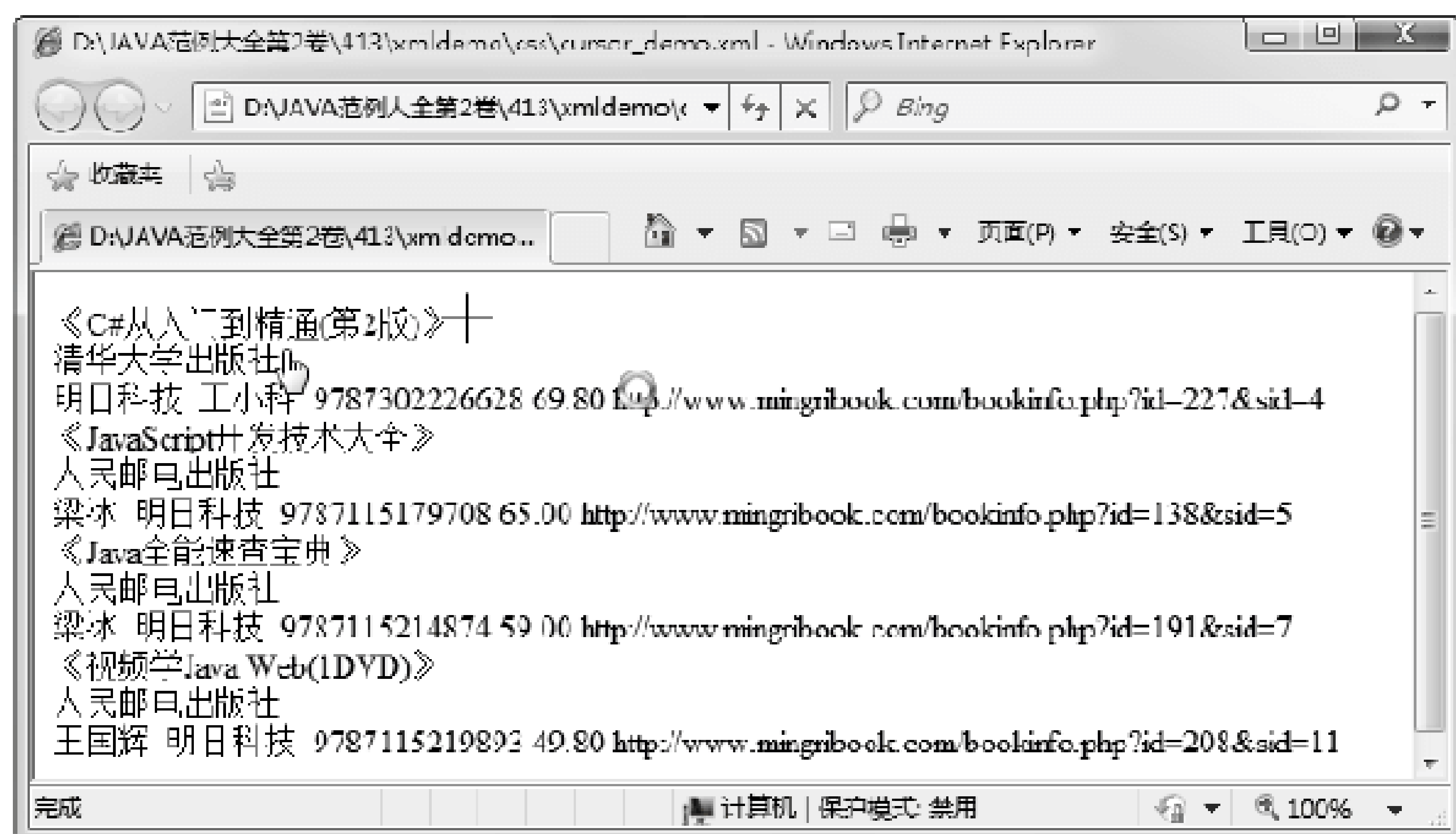


图 1.2　CSS 改变 XML 中鼠标指针形状

关键技术

在 CSS 样式中，对超链接的样式有以下几种定义。

（1）设置链接未被访问时的样式，具体写法如下：

```
a:link{font-size:10px;... }
```

（2）设置链接在鼠标经过时的样式，具体写法如下：

```
a:hover{font-size:10px;text-decoration:underline;color:#ff0000}
```

（3）设置链接激活时的样式，具体写法如下：

```
a:active{font-size:10px;...}
```

（4）设置链接已被访问过的样式，具体写法如下：

```
a:visited{font-size:10px;color:#00ffff;...}
```

设计过程

（1）新建 index.html 页，在该页中首先定义超链接的 CSS 样式。关键代码如下：

```
<head>
	<title>超链接的样式</title>
	<--设置超链接的样式-->
		<meta http-equiv="pragma" content="no-cache">
		<meta http-equiv="cache-control" content="no-cache">
		<meta http-equiv="expires" content="0">
		<style type="text/css">
			td{
				font-size:9pt;
				color:#007766;
			}
			a:link{
				font-size:9pt;
				color:#cccccc;
				font-weight:bold;
				text-decoration: underline;
			}
			a:hover{
				font-size:9pt;
				color:#ff0000;
				font-weight:bold;
				text-decoration: underline;
			}
			a:active{
				font-size:9pt;
				color:#6699ff;
				font-weight:bold;
				text-decoration: underline;
			}
		</style>
</head>
```

（2）在该页中，使用<a>标签添加超链接。关键代码如下：

```
<table align="center">
<--创建表格-->
	<tr>
		<td>明日科技→<a href="#">明日科技</a></td>
		<--添加超链接-->
	</tr>
</table>
```


心法领悟 002：CSS 样式中的超链接。

超链接是每个网站中必有的功能，换句话说，它是网站中最常用的功能。因此，有必要掌握在 CSS 样式中超链接的几个属性（a:link、a:hover、a:active、a:visited）的应用。

实例 003　CSS 在 XML 中添加背景图

光盘位置：光盘\MR\01\003

初级

实用指数：★★★★

实例说明

在显示 XML 文档内容时，如只是像实例 001 中那样简单地格式化，从视觉上难免会觉得有些单调。如果为它加上背景图，无疑会显得更美观。本实例将实现使用 CSS 为 XML 文档添加背景图，使文档显示效果更加美观。实例运行结果如图 1.3 所示。

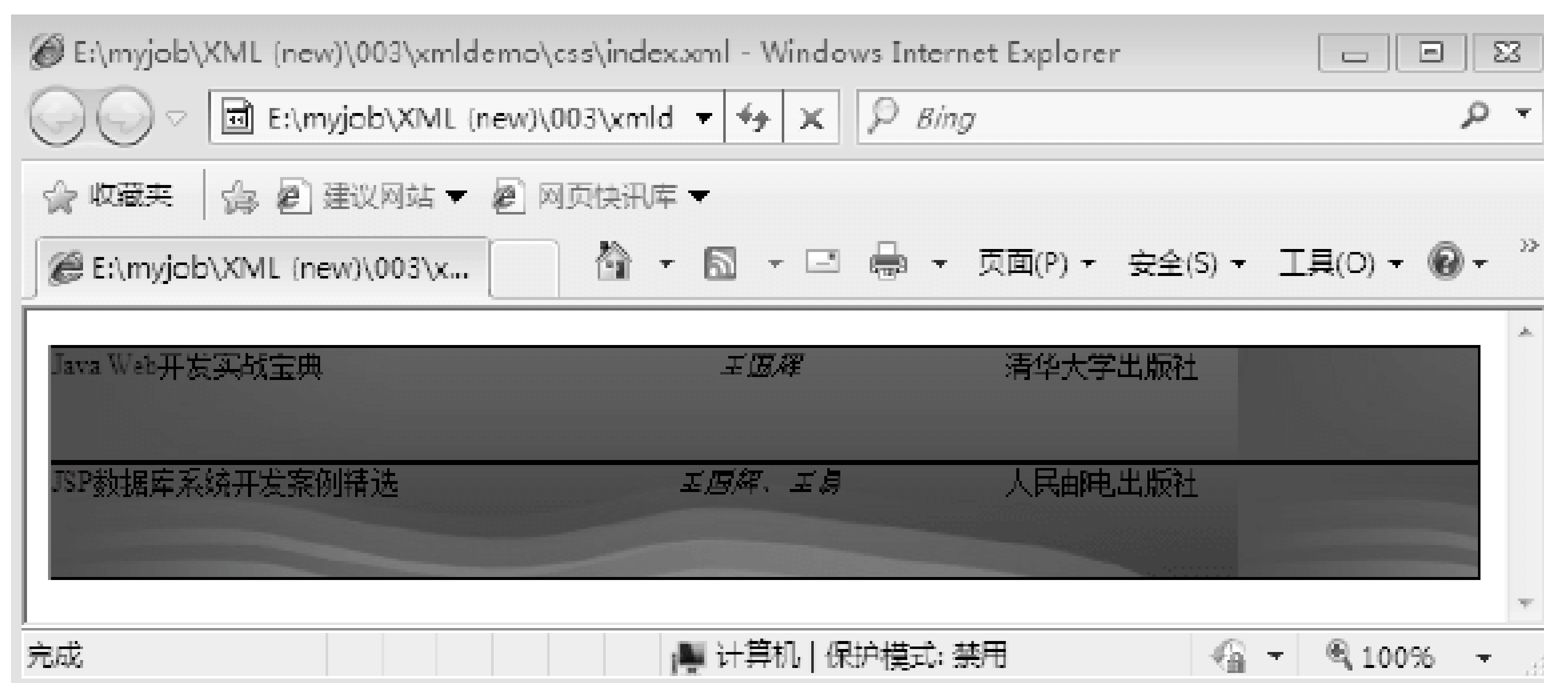


图 1.3　CSS 在 XML 中添加背景图

关键技术

本实例的实现技术和实例 001 类似，区别在于加入了设置背景图的属性 background-image。使用 background-image 属性的语法格式如下：

```
books{
background-image:url(001.jpg);          //设置 books 元素的背景图为 001.jpg
}
```

（1）编写 CSS 样式表文件 style.css，在该文件中指定 XML 文档中各元素的显示样式，加入添加元素背景图的属性 background-image。代码如下：

```
books{
background-image:url(001.jpg);
<--添加背景图片-->
}
book{
border:1px solid;
width:100%;
height:50px;
font-size:12px;
}
ISBN{
display: none;
}
bookname{
width: 200px;
}
author{
font-style: italic;
width: 200px;
text-align:center;
}
publishing{
width: 100px;
}
```

（2）编写 index.xml 文件，在该文件中创建一个 books 根元素，该元素由多个 book 元素组成。代码如下：

```
<?xml version="1.0" encoding="gb2312" ?>
<!--声明 XML 文档版本与字符编码方式-->
<books>
<--创建根元素-->
<book>
        <ISBN>7-302-21033-7</ISBN>
        <bookname>Java Web 开发实战宝典</bookname>
        <author>王国辉</author>
        <publishing>清华大学出版社</publishing>
</book>
<book>
        <ISBN>7-115-14689-6</ISBN>
        <bookname>JSP 数据库系统开发案例精选</bookname>
        <author>王国辉、王易</author>
        <publishing>人民邮电出版社</publishing>
</book>
</books>
```

（3）在 index.xml 文件的第 2 行加入如下引用 CSS 样式表文件的代码。

```
<?xml-stylesheet href="style.css" type="text/css"?>
<--引入 CSS 样式表-->
```

秘笈心法

心法领悟 003：HTML 中的背景图。

在实现 HTML 语言时也可以这样做，以提高网页的可维护性。

实例 004	CSS 制作 XML 表格 光盘位置：光盘\MR\01\004	初级 实用指数：★★★★

实例说明

XML 本身规则性很强，实际应用中经常会把 XML 的数据以表格的形式展示出来，通过 CSS 把 XML 格式转换成表格也是一种方式，如图 1.4 所示就是使用 CSS 把 XML 格式化成一个简单的表格。

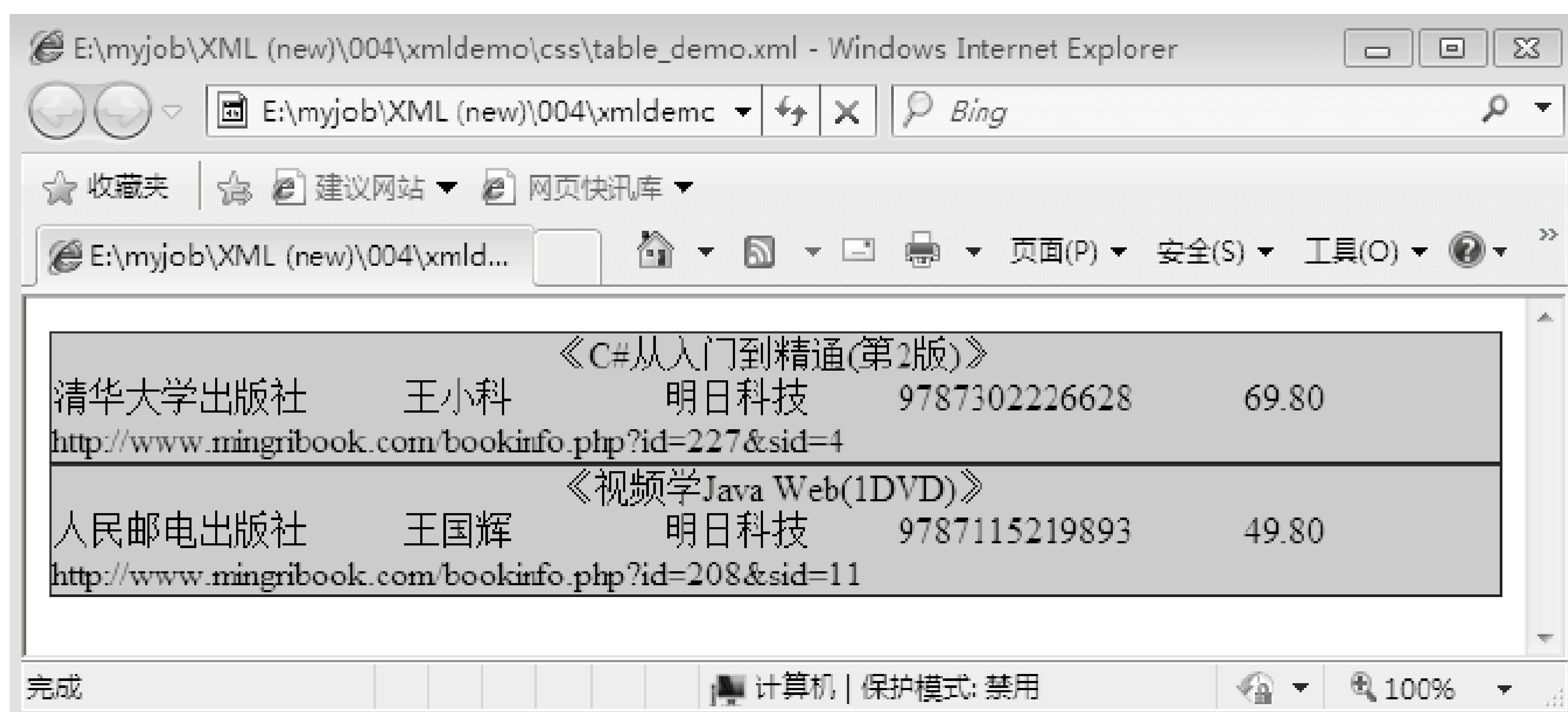


图 1.4 CSS 制作 XML 表格

通过 CSS 制作 XML 表格需要综合运用 CSS 多项属性来共同完成，各种属性作用于各个 XML 元素之上，才会展示出希望达到的效果。代码如下：

```
<--指定各元素样式-->
book
{
display: block;
        border-width: 1px;
        border-color: #930;
        border-style:outset;
```

```
        background-color: #CCC
}
name
{
        display: block;
        text-align: center;
}
publisher,company,author,ISBN,price
{
        text-align: center;
        width: 18%
}
url
{
        display: block;
}
```

（1）建立 XML 文件。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<?xml-stylesheet type="text/css" href="table_demo.css"?>
<books>
<--创建根元素-->
        <book>
        <--创建子元素及内容-->
                <name>《C#从入门到精通(第 2 版)》</name>
                <publisher>清华大学出版社</publisher>
                <author>王小科</author>
                <company>明日科技</company>
                <ISBN>9787302226628</ISBN>
                <price unit="RMB" >69.80</price>
                <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
        </book>
        <book>
        <--创建子元素及内容-->
                <name>《视频学 Java Web(1DVD)》</name>
                <publisher>人民邮电出版社</publisher>
                <author>王国辉</author>
                <company>明日科技</company>
                <ISBN>9787115219893</ISBN>
                <price unit="RMB" >49.80</price>
                <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
        </book>
</books>
```

（2）引入 CSS 后，先在 book 元素上使用 display:block 属性，让 book 元素按 bookCSS 样式模板排开；然后对 book 的边框进行修饰，如添加边框样式、设置边框颜色、调整边框宽度等。代码如下：

```
book
<--指定 book 元素的显示样式-->
{
display: block;
        border-width: 1px;
        border-color: #930;
        border-style:outset;
        background-color: #CCC
}
```

（3）完成 book 样式编辑后，将 name 元素加上 display: block 属性使 name 单独成一行，并将 name 的文本居中显示。代码如下：

```
name
<--指定 name 元素的样式-->
{
        display: block;
        text-align: center;
}
```

（4）把 publisher、company、author、ISBN、price 元素居中显示，同时设置其宽度。代码如下：

```
publisher,company,author,ISBN,price
<--指定各元素的样式-->
{
        text-align: center;
        width: 18%
}
```

（5）url 的内容比较多，如果和 publisher 等元素放在一行显示会影响美观，所以将 url 元素加上 display:block 属性使其独立在一行上显示。代码如下：

```
url
{
        display: block;
}
```

秘笈心法

心法领悟 004：CSS 样式中制作表格的元素区分。

使用 CSS 制作表格时，如果多个元素连在一行显示，则很难把每个元素区分开来。对此，可以在显示时让各元素之间保持一定的距离。该距离可以使用相对值（如 width:18%）的方式来控制，也可以使用 px 和 p 控制。

实例 005 XML 中提取节点字符串值

高级

光盘位置：光盘\MR\01\005

实用指数：★★★★

实例说明

使用 XSLT 转换 XML，最基本的就是把 XML 文档的内容取出来显示在浏览器中，这时就需要用到 <xsl:value-of />，如图 1.5 所示为使用 value-of 取出的值。

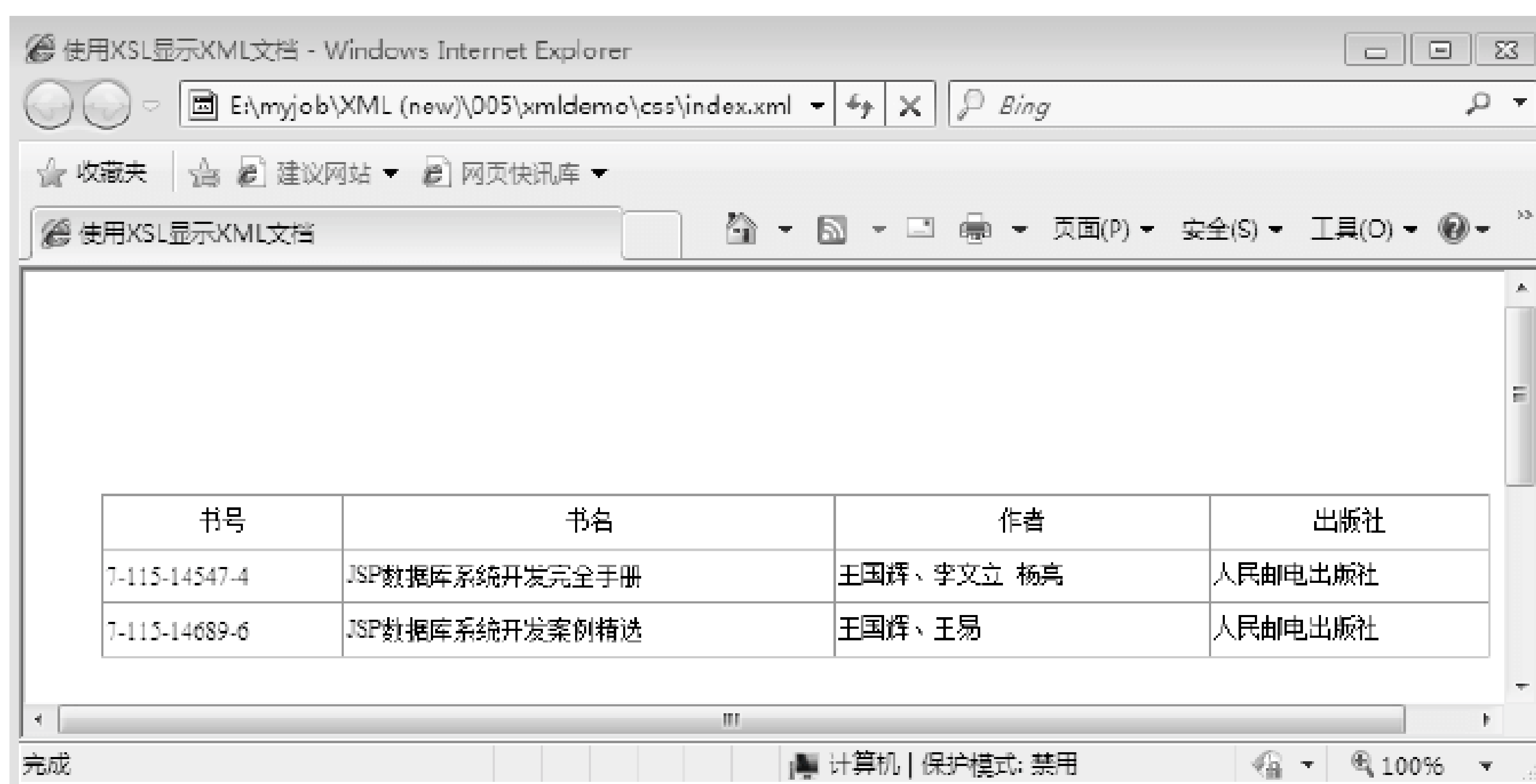


图 1.5　XML 中提取节点字符串值

##

<xsl:value-of />中的必选属性 select 根据 XML 的元素节点取值显示内容，每级节点之间使用“/”分开。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
  <body>
     <table border="1">
 <--创建表格-->
        <tr>
           <td><xsl:value-of select="books/book/name"/></td>
```

```
            <td><xsl:value-of select="books/book/publisher"/></td>
            <td><xsl:value-of select="books/book/company"/></td>
            <td><xsl:value-of select="books/book/author"/></td>
            <td><xsl:value-of select="books/book/ISBN"/></td>
            <td><xsl:value-of select="books/book/price"/></td>
          </tr>
        </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

（1）编写 XSL 样式表文件 style.xsl，在该文件中将 XML 的<book>标签转换为表格的 TR 标记，<book>标签下的子元素转换为表格的 TD 标记，并将整个文档作为 HTML 输出。代码如下：

```
<?xml version='1.0' encoding='gb2312'?>
<!--声明 XML 文档的版本及编码方式-->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head>
<title>使用 XSL 显示 XML 文档</title>
<style>
td {
font-size: 9pt; color: #000000;
}
body {
margin: 0px;
}
.tableBorder {
border: #aaaaaa 1px solid
}
</style>
</head>
<body>
<table width="780"   border="0" align="center" cellpadding="0" cellspacing="0"
background="Images/bg.jpg" class="tableBorder">
<tr>
<td width="26%" height="112" valign="top"> </td>
</tr>
<tr>
<td height="248" align="center" valign="top"><table width="90%" border="1"
bordercolor="#FFFFFF" bordercolordark="#FFFFFF" bordercolorlight="#999999"
cellpadding="0" cellspacing="0">
<tr height="27">
<td align="center">书号</td>
<td align="center">书名</td>
<td align="center">作者</td>
<td align="center">出版社</td>
</tr>
<xsl:for-each select="books/book">
<tr height="27">
<td><xsl:value-of select="ISBN"/></td>
<td><xsl:value-of select="bookname"/></td>
<td><xsl:value-of select="author"/></td>
<td><xsl:value-of select="publishing"/></td>
</tr>
</xsl:for-each>
</table></td>
</tr>
<tr>
<td height="69" valign="top"> </td>
</tr>
</table>
</body>
</html>
</xsl:template>
```

```
</xsl:stylesheet>
```

（2）编写 index.xml 文件，在该文件中创建一个 books 根元素，该元素由多个 book 元素组成。代码如下：

```
<?xml version="1.0" encoding="gb2312" ?>
<!--声明 XML 文档版本与字符编码方式-->
<books>
        <book>
                <ISBN>7-115-14547-4</ISBN>
                <bookname>JSP 数据库系统开发完全手册</bookname>
                <author>王国辉、李文立  杨亮</author>
                <publishing>人民邮电出版社</publishing>
        </book>
<book>
<ISBN>7-115-14689-6</ISBN>
<bookname>JSP 数据库系统开发案例精选</bookname>
<author>王国辉、王易</author>
<publishing>人民邮电出版社</publishing>
</book>
</books>
```

（3）在 index.xml 文件的第 2 行加入以下引用 XSL 样式表文件的代码。

```
<?xml-stylesheet href="style.xsl" type="text/xsl"?>
```

秘笈心法

心法领悟 005：控制浏览器转义。

在使用<xsl:value-of>时，如果不希望浏览器对取出的内容进行转义，则应设置 disable-output-escaping="yes"，否则使用 disable-output-escaping="no"。例如，如果设置 disable-output-escaping="yes"，则页面中会直接显示“<”符号而不进行转换。如果设置 disable-output-escaping="no"，则 XSLT 会把“<”转换成“<”显示出来。代码如下：

```
private static void move(char x,char y){
        System.out.printf("%c-->%c",x,y);
        System.out.print("\n");
}
//将 n 个盘子从第 1 座借助第 2 座移到第 3 座
private static void hanoit(int n,char one,char two,char three){
        if(n==1){                              //如果只有一个盘子
                move(one,three);
        }
        else{
                hanoit(n-1,one,three,two);     //将第 1 座上的盘子借助第 3 座移到第 2 座上
                move(one,three);
                hanoit(n-1,two,one,three);     //将第 2 座上的盘子借助第 1 座移到第 3 座上
        }
}
public static void main(String[] args) {
        int m;
        System.out.println("请输入你要移动的盘子数：");
        Scanner s=new Scanner(System.in);
        m=s.nextInt();
        System.out.println("移动"+m+"个盘子的步骤如下");
        hanoit(m,'A','B','C');
}
```

实例 006 在 XML 内部定义 DTD

高级

光盘位置：光盘\MR\01\006

实用指数：★★★★★

XML 的使用规则是由使用者自己定义的，其他用户如果想使用这个 XML 就要遵守其使用规则。该规则可以通过 DTD（Document Type Definition，文档类型定义）来定义。本实例使用 DTD 定义了一个 XML 文档结构，

并且在 XML 中使用了该 DTD，运行结果如图 1.6 所示。

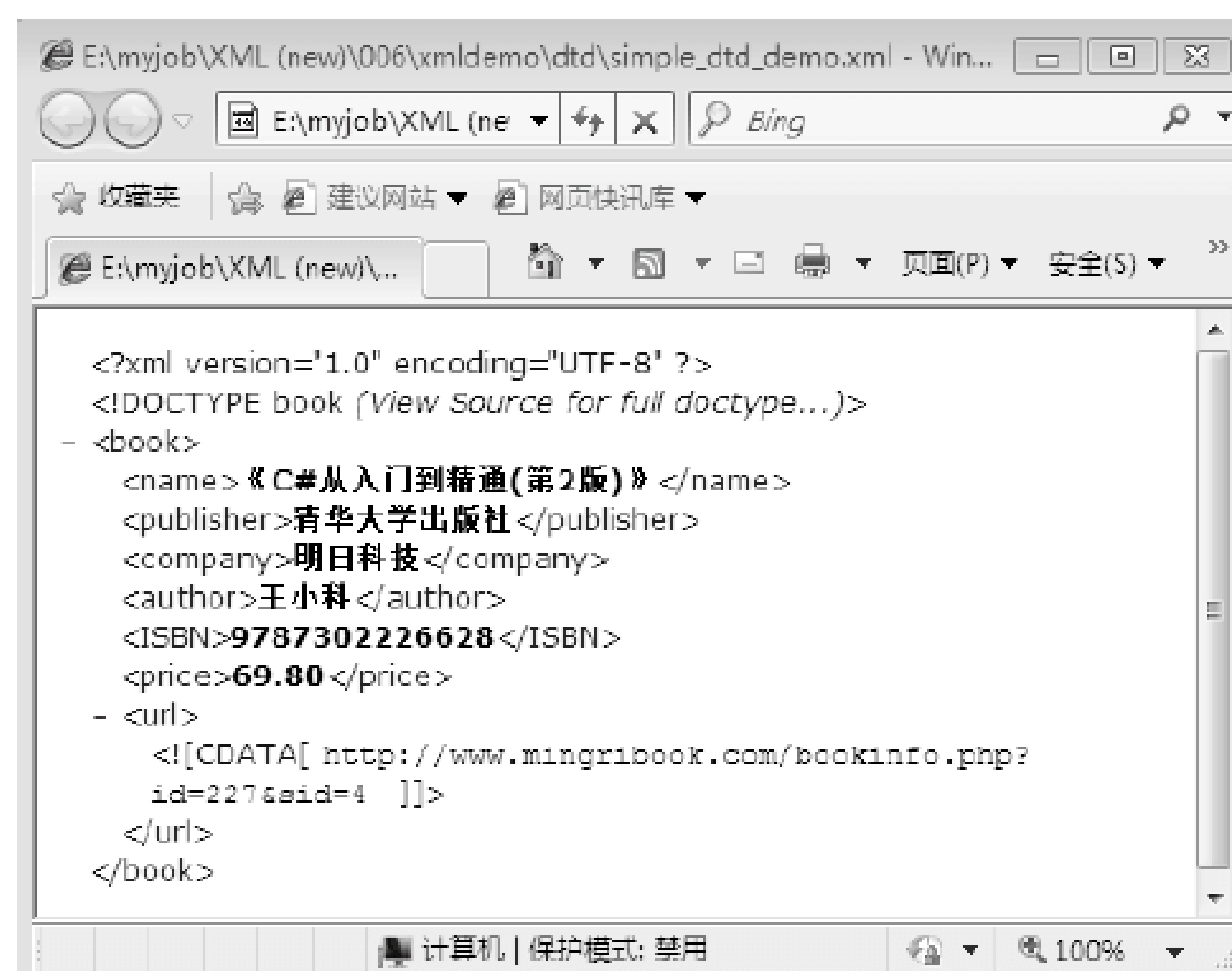


图 1.6　在 XML 内部定义 DTD

关键技术

在定义 XML 文档结构时，要先使用 DOCTYPE 声明 DTD；声明完 DTD 后，再根据 DTD 定义的内容编写 XML 代码。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档的版本及编码方式-->
<!DOCTYPE    book [
<--声明 DTD 文档-->
        <!ELEMENT book (name)>
        <!ELEMENT name (#PCDATA)>
]>
<book>
        <name>《C#从入门到精通(第 2 版)》</name>
</book>
```

（1）建立一个空的 XML 文档。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档的版本及编码方式-->
```

（2）先根据需求构思 XML 文档的格式，声明 DTD 内容。代码如下：

```
<!DOCTYPE    book [
<--声明 DTD 文档-->
        <!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
        <!ELEMENT name          (#PCDATA)>
        <!ELEMENT publisher     (#PCDATA)>
        <!ELEMENT company       (#PCDATA)>
        <!ELEMENT author        (#PCDATA)>
        <!ELEMENT ISBN          (#PCDATA)>
        <!ELEMENT price         (#PCDATA)>
        <!ELEMENT url           (#PCDATA)>
]>
```

（3）根据 DTD 的定义情况创建 XML 文档内容。代码如下：

```
<book>
<--创建元素 book-及内容->
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price>69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 006：验证 DTD 错误。

一般在编写 XML 时，最简单的方法是通过浏览器打开 XML 验证编写是否有书写上的错误。在 XML 中定义了 DTD 后，同样可以使用这种方法验证 DTD 是否有拼写等错误。

实例 007 在 XML 外部引用 DTD

光盘位置：光盘\MR\01\007

高级

实用指数：★★★★★

实例说明

使用 DTD 时可以在 XML 文档内部定义，但为了引用方便和书写规范，一般都把 DTD 单独定义成一个 DTD 文档。XML 可以通过引用的方式使用 DTD 的定义，如图 1.7 所示便是一个引用外部 DTD 的 XML 文档。

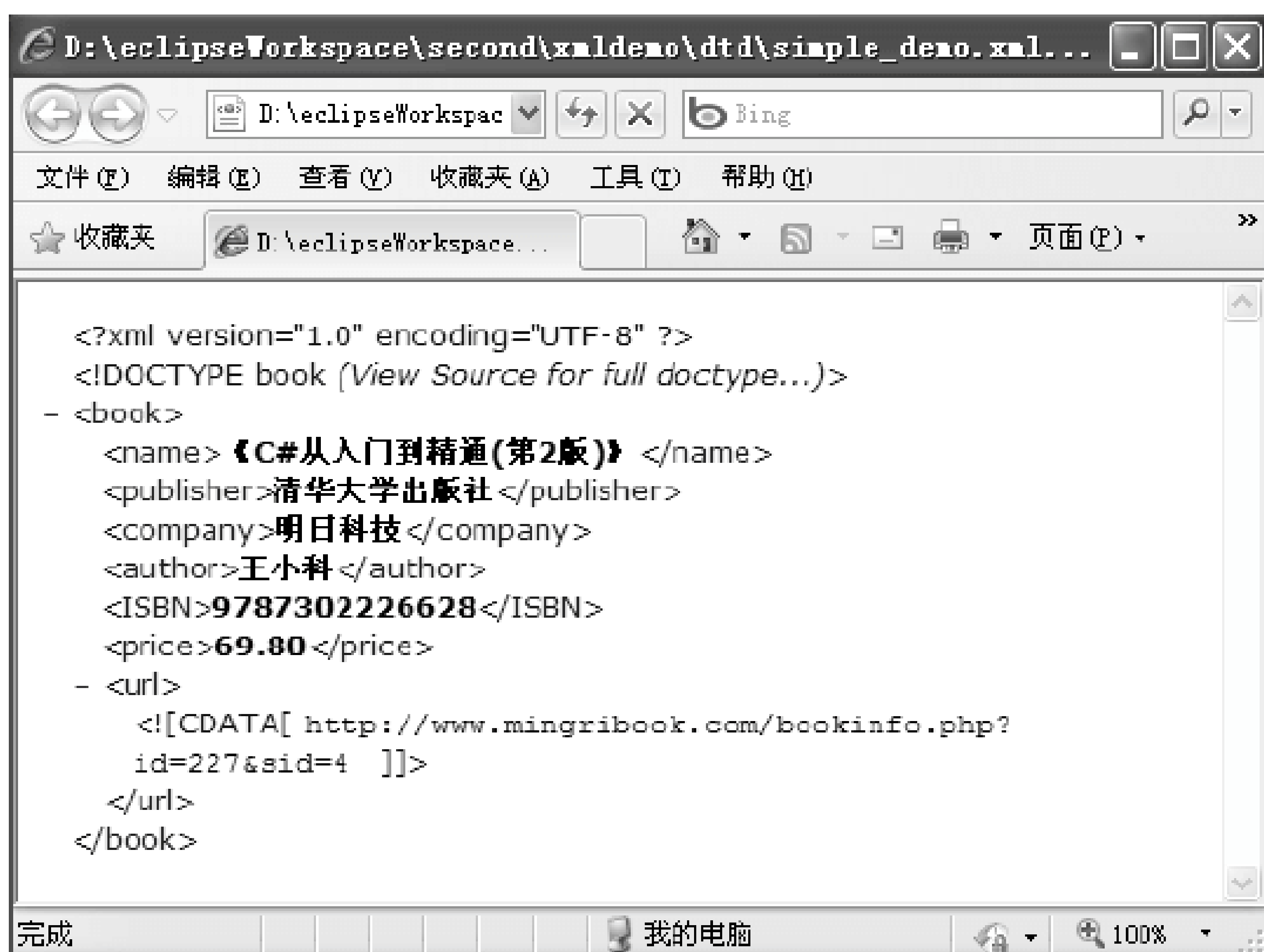


图 1.7　在 XML 外部引用 DTD

关键技术

DOCTYPE 表示声明了一个 DTD，book 表示 XML 的根节点，SYSTEM 表示引用一个外部的 DTD，在 SYSTEM 后面填写 DTD 的 URL 地址，这样就在 XML 中引用了一个 DTD 文档。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档的版本及编码方式-->
<!DOCTYPE book SYSTEM "simple_demo.dtd">
<!--引入外部 DTD 文件-->
<book>
        ……//省略部分代码
</book>
```

（1）先根据需求构思 XML 文档的格式，建立 DTD 文档。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!--定义 DTD 文档-->
<!ELEMENT name          (#PCDATA)>
<!ELEMENT publisher     (#PCDATA)>
<!ELEMENT company       (#PCDATA)>
<!ELEMENT author        (#PCDATA)>
<!ELEMENT ISBN          (#PCDATA)>
<!ELEMENT price         (#PCDATA)>
```

```
<!ELEMENT url      (#PCDATA)>
```

（2）根据 DTD 的定义创建一个 XML 文档。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档的版本及编码方式-->
```

（3）在创建好的 XML 文档中引入 DTD 文档。代码如下：

```
<!DOCTYPE book SYSTEM "simple_demo.dtd">
<!--引入 DTD 文档-->
```

秘笈心法

心法领悟 007：特殊的 DTD。

在 XML 中引用 DTD 时，首先使用 DOCTYPE 声明，然后定义 XML 文档中的根元素，接着使用 SYSTEM 表示 XML 需要引用的外部 DTD 资源。SYSTEM 的位置既可以是 SYSTEM，也可以是 PUBLIC。当定义为 PUBLIC 时，表示引用的这个 DTD 是由权威机构制定的，提供给特定行业或公众使用的。

实例 008　验证 XML 是否符合 DTD 的定义　高级

光盘位置：光盘\MR\01\008　实用指数：★★★★★

实例说明

使用 DTD 可以定义 XML 的文档结构，但是该 XML 文档结构是否遵守了 DTD 的定义规则，可以通过程序验证 XML 是否符合 DTD 的定义来判断。如图 1.8 所示是使用 IE 验证 XML 是否符合 DTD 的定义。



图 1.8　验证 XML 是否符合 DTD 的定义

关键技术

声明一个 XML DOM 对象，使用 load()方法加载需要验证的 XML 文档。解析器在加载 XML 时也会一起加载 DTD 文档，同时也会完成验证工作。如果解析器验证没有错误，则 xmlDoc.parseError.errorCode 的值为 0；如果有错误，则 xmlDoc.parseError.errorCode 的值是与之相对应的错误值。同时，使用 xmlDoc.parseError. reason 和 xmlDoc.parseError.line 还可以准确地输出错误的原因和错误的行数。

（1）先设计一个 HTML 文档，添加一个 text 标签接收需要验证的 XML。具体代码如下：

```
<center>
<!--声明 XML 文档居中显示-->
<h2>通过 DTD 验证 XML 文件</h2>
<!--声明 XML 文档的标题-->
<p>
<input id="xmlfile" type="text" />
</p>
</center>
```

（2）添加一个 button 按钮，在该按钮上添加 onClick()方法。这样每次单击该按钮时，都将触发验证 XML 文档的 JavaScript 方法。具体代码如下：

```
<input type="button" name="submit" value="验　证" onClick="validateXML(document.getElementById('xmlfile').value);" />
<!--添加按钮并声明 onclick()方法-->
```

（3）添加一个 id 为 showError 的 div 标签，用于显示输出信息到页面上。具体代码如下：

```
<div id="showError"></div>
<!--添加 div 标签-->
```

（4）完成验证 DTD 的方法，在方法中把错误的信息输出到 showError 的位置上。代码如下：

```
function validateXML(filename){
        var txt="";
        if (window.ActiveXObject){
                document.getElementById("showError").innerText ="";
                var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
                xmlDoc.async = false;
                xmlDoc.validateOnParse= true;
                xmlDoc.load(filename);
                if(xmlDoc.parseError.errorCode!=0){
                txt="Error Code: " + xmlDoc.parseError.errorCode + "\n";
                txt=txt+"Error Reason: " + xmlDoc.parseError.reason ;
                txt=txt+"Error Line: " + xmlDoc.parseError.line;
              }else{
                txt="没有错误";
              }
                document.getElementById("showError").innerText = txt;
        }else{
                        alert("此浏览器不支持验证!");
}
}
</script>
<script type="text/javascript">
```

秘笈心法

心法领悟 008：innerText()和 innerHtml()的区别。

在输出 DTD 的错误信息时先获取 showError 标签，然后在页面中把错误信息输出到 showErro 的标签上。在此可以使用 innerText()和 innerHtml()两种方法，两者的区别在于 innerText()可以把信息完整地输出到页面上，innerHtml()则会把信息按照 HTML 语言的方式输出到页面上。例如，在页面中输出一个
时，innerText()输出以后就是
字样；而使用 innerHtml()输出时，页面上只会输出一个回车。

实例 009 在 DTD 中声明元素

高级

光盘位置：光盘\MR\01\009

实用指数：★★★★★

XML 的基本单位是元素，所以在 DTD 中声明元素也是最基本的。ELEMENT 用于 DTD 元素的声明（在声明的同时还可以定义元素的各种使用情况）。实例运行结果如图 1.9 所示。

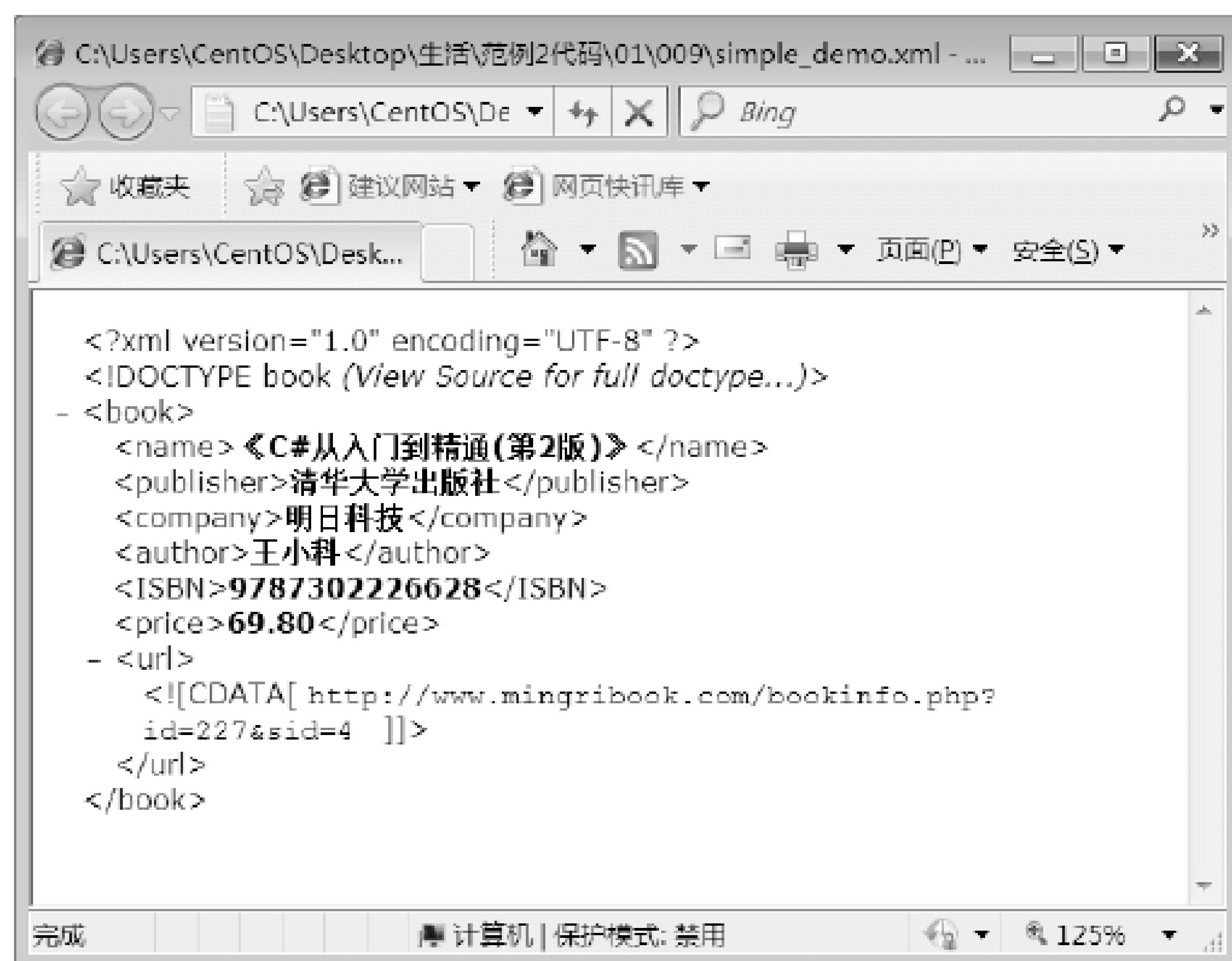


图 1.9　在 DTD 中声明元素

关键技术

使用 ELEMENT 声明一个元素 book。book 是 XML 的根元素，其下包含 name、publisher、company、author、ISBN、price、url 7 个子元素，使用括号把这 7 个子元素括起来即可完成根元素的声明。接下来，分别声明这 7 个子元素，其内容是#PCDATA 类型。

设计过程

（1）建立一个 DTD 文档，在其中使用 ELEMENT 定义根元素及其内容。具体代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--引入外部 DTD 文件-->
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!--定义根元素及其子元素-->
```

（2）根据上面定义的根元素的内容，分别定义各子元素，同时还要定义其内容。具体代码如下：

```
<!--定义子元素的内容-->
<!ELEMENT name          (#PCDATA)>
<!ELEMENT publisher     (#PCDATA)>
<!ELEMENT company       (#PCDATA)>
<!ELEMENT author        (#PCDATA)>
<!ELEMENT ISBN          (#PCDATA)>
<!ELEMENT price         (#PCDATA)>
<!ELEMENT url           (#PCDATA)>
```

（3）根据定义的 DTD 文档，可以建立相应的 XML 文件。具体代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<!DOCTYPE book SYSTEM "simple_demo.dtd">
<!--引入外部 DTD 文件-->
<book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price>69.80</price>
        <url>
        <![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]>
        </url>
</book>
```

秘笈心法

心法领悟 009：ANY 属性的使用。

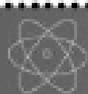
在定义 DTD 时，如果想使某一元素可以包含任意的内容，则可以把元素内容定义成 ANY。例如，上面定义 book 元素只能包含 name、publisher、company、author、ISBN、price、url 这 7 个子元素，如果想使其包含任意内容，可以这样定义：

```
<!ELEMENT book ANY>
```

如果希望元素内容没有任何值，可以定义为 EMPTY。

```
<!ELEMENT book EMPTY>
```

实例 010	在 DTD 中声明重复元素 光盘位置：光盘\MR\01\010	高级 实用指数：★★★★★



在定义 DTD 时，通常是在父元素的内容里声明子元素的名称。不加任何条件时，该子元素在父元素里只能出现一次，如果在 XML 中定义多个子元素，使用验证工具进行检验时将出现如图 1.10 所示错误。



图 1.10　在 DTD 中声明重复元素

关键技术

如果希望父元素内部能出现多个子元素，声明时要在子元素后面添加“+”。例如，一本书可能有多个作者，定义父元素时，在父元素内部的 author 后面添加“+”。

设计过程

（1）建立一个 DTD 文档，在其中使用 ELEMENT 定义根元素及其内容（在可以重复的子元素后添加“+”）。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<!ELEMENT book (name,publisher,company,author+,ISBN,price,url)>
<!--定义根元素及其子元素-->
```

（2）根据上面定义根元素的内容，分别定义各子元素及其内容。代码如下：

```
<!--定义子元素及其内容-->
<!ELEMENT name          (#PCDATA)>
<!ELEMENT publisher     (#PCDATA)>
<!ELEMENT company       (#PCDATA)>
<!ELEMENT author        (#PCDATA)>
<!ELEMENT ISBN          (#PCDATA)>
<!ELEMENT price         (#PCDATA)>
<!ELEMENT url           (#PCDATA)>
```

（3）根据定义的 DTD 文档，可以建立相应的 XML 文件。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<!DOCTYPE book SYSTEM "author_demo.dtd">
<!--引入外部 DTD 文件-->
<book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <author>徐薇</author>                    //添加重复的元素
        <ISBN>9787302226628</ISBN>
        <price>69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

心法领悟 010：子元素出现次数的控制。

使用“+”表示 XML 子元素在其父元素中至少要出现一次，同时可以出现多次。这就说明子元素在父元素中是必须出现的。如果在使用 XML 时允许子元素在父元素中不出现，也可以出现多次，则应在子元素后添加“*”。例如：

```
<!ELEMENT book (name,publisher,company,author*,ISBN,price,url)>
```

在元素后面使用“?”，表示该元素可以不出现，如果出现也只能出现一次。例如：

```
<!ELEMENT book (name,publisher,company,author?,ISBN,price,url)>
```

实例 011　在 DTD 中声明选择性元素

光盘位置：光盘\MR\01\011

高级

实用指数：★★★★★

实例说明

为了满足 XML 数据的多样性，DTD 提供了多种声明方式。有这样一种情况，XML 父元素中有多个子元素，但是有两个子元素必须在父元素里出现一个，而且不能同时出现。例如，在 book 元素里必须要有 tel 或者 phone 两个子元素，但是这两个子元素不能同时出现，XML 中要么使用 tel，要么使用 phone。如果没有按 DTD 的声明使用 XML，验证时就会出现错误，如图 1.11 所示。



图 1.11　在 DTD 中声明选择性元素

关键技术

tel 和 phone 属于 book 的子元素，两者必须有一个出现，但不能同时出现。book 中的 name、publisher、company、author、ISBN、price、url 和(tel|phone)是并列关系。

（1）建立一个 DTD 文档，在其中使用 ELEMENT 定义根元素及其内容。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<!ELEMENT book (name,publisher,company,author,ISBN,price,url,(tel|phone))>
<!--定义根元素及其子元素-->
```

（2）根据上面定义的根元素内容，分别定义各子元素及其内容。tel 和 phone 虽然不能同时出现在 XML 父元素中，但是在 DTD 中二者都要事先声明。具体代码如下：

```
<!--定义子元素及其内容-->
<!ELEMENT name          (#PCDATA)>
<!ELEMENT publisher     (#PCDATA)>
<!ELEMENT company       (#PCDATA)>
<!ELEMENT author        (#PCDATA)>
<!ELEMENT ISBN          (#PCDATA)>
<!ELEMENT price         (#PCDATA)>
<!ELEMENT url           (#PCDATA)>
<!ELEMENT tel           (#PCDATA)>
<!ELEMENT phone         (#PCDATA)>
```

（3）根据定义的 DTD 文档，可以建立相应的 XML 文件。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<!DOCTYPE book SYSTEM "tel_demo.dtd">
<!--引入外部 DTD 文件-->
<book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price>69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
        <tel>13812345678</tel>
</book>
```

秘笈心法

心法领悟 011：“|”和“*”混合使用。

在 DTD 中把“|”和“*”混合在一起使用，可使定义的元素更灵活，例如：

```
<!ELEMENT book (name|publisher|company|author|ISBN|price|url)*>
```

上述代码表示 book 中的子元素可以在任意位置出现，并且没有次数的限制。这种 DTD 声明让用户在使用 XML 时更自由。

实例 012　在 DTD 中使用 ENTITY

光盘位置：光盘\MR\01\012

高级

实用指数：★★★★★

实例说明

使用 ENTITY 可以声明一个实体，而在 XML 中可以通过引用实体名称实现对 ENTITY 的调用，进而实现重用公共内容，提高代码效率。这对于可能出现多次的内容来讲用途很大。本实例运行结果如图 1.12 所示。

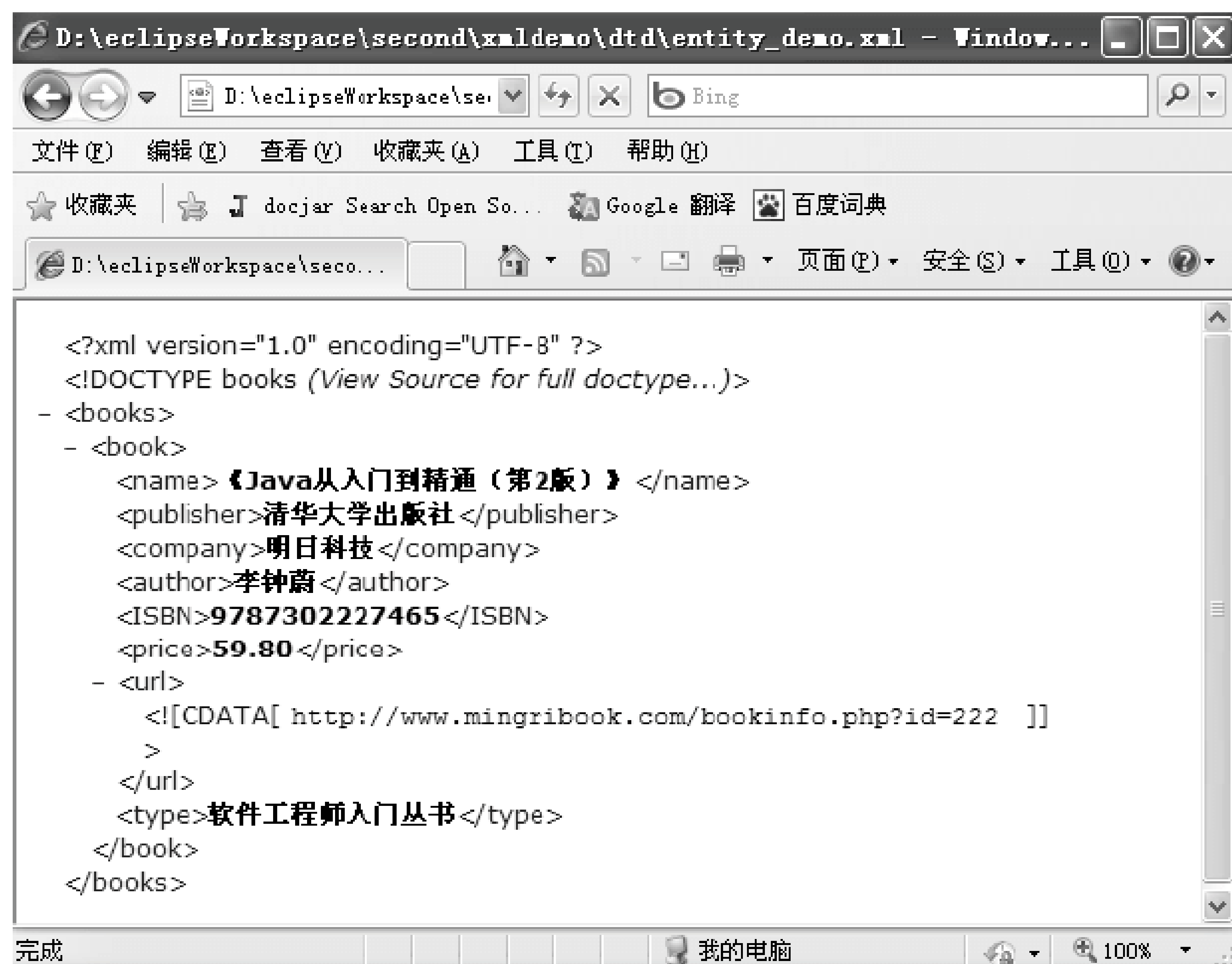


图 1.12　在 DTD 中使用 ENTITY

关键技术

声明实体时使用 ENTITY 关键字。通常在 ENTITY 关键字后面声明实体名称 info，接着声明实体内容“软件工程师入门丛书”。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<!ELEMENT books (book+)>
<!--定义根元素及规则-->
<!ELEMENT book     (name,publisher,company,author,ISBN,price,url,type)>
……//省略部分代码
<!ENTITY info "软件工程师入门丛书" >
```

（1）使用 ELEMENT 定义根元素 books。books 含有子元素 book，由于允许 book 在 books 中出现多次，所以 book 后面要添加“+”，如 (book+)。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
```

```
<!ELEMENT books (book+)>
<!--定义根元素 books-->
```

（2）使用 ELEMENT 为 book 声明其子元素 name、publisher、company、author、ISBN、price、url 和 type，并为每个子元素设定元素类型。代码如下：

```
<!ELEMENT book      (name,publisher,company,author,ISBN,price,url,type)>
```

（3）使用 ENTITY 声明实体名称 info，在 info 后面定义实体内容“软件工程师入门丛书”。具体代码如下：

```
<!--定义子元素及其内容-->
<!ELEMENT name          (#PCDATA)>
<!ELEMENT publisher     (#PCDATA)>
<!ELEMENT company       (#PCDATA)>
<!ELEMENT author        (#PCDATA)>
<!ELEMENT ISBN          (#PCDATA)>
<!ELEMENT price         (#PCDATA)>
<!ELEMENT url           (#PCDATA)>
<!ELEMENT type          (#PCDATA)>
<!ENTITY info "软件工程师入门丛书" >
```

（4）在 XML 文档中使用实体时要在实体前后添加“&”和“;”，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<!DOCTYPE books SYSTEM "entity_demo.dtd">
<!--引入外部 DTD 文件-->
<books>
<!--创建根元素-->
        <book>
        <!--创建子元素及内容-->
                <name>《Java 从入门到精通（第 2 版）》</name>
                <publisher>清华大学出版社</publisher>
                <company>明日科技</company>
                <author>李钟蔚</author>
                <ISBN>9787302227465</ISBN>
                <price>59.80</price>
                <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=222]]></url>
                <type>&info;</type>
        </book>
</books>
```

（5）引用实体后，在 XML 文档被解析时会直接调用实体内容。

秘笈心法

心法领悟 012：引用外部实体。

本例声明的是内部实体，在其他应用中还可以使用 ENTITY 声明外部实体。在引用外部实体时必须添加关键字 SYSTEM 或者 PUBLIC，例如：

```
<!ENTITY info SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >
```

1.2　应用 XML Schema

Schema 是使用 XML 语法编写的，与 DTD 相比，它更易于学习和使用。下面具体介绍其应用。

实例 013	验证 XML 是否符合 Schema 的描述 光盘位置：光盘\MR\01\013	初级 实用指数：★★★★

定义完 Schema 以后，开发人员可以根据定义的 Schema 建立 XML 文档。在编写 XML 文档时，很可能由于拼写错误导致在使用 XML 时出现错误。本例将编写一个程序，通过 IE 浏览器验证 XML 是否符合 Schema

的定义，运行结果如图 1.13 所示。



图 1.13　验证 XML 文档是否符合 Schema 的定义

■

首先声明一个 MSXML2.DOMDocument.4.0 对象，然后使用 load()方法加载需要验证的 XML 文档。在加载 XML 时，系统会自动加载 XSD 文件，同时完成验证工作。如果验证通过（即没有错误），xmlDoc.parseError.errorCode 的值为 0；如果验证没有通过（即有错误），xmlDoc.parseError.errorCode 的值就是相对应的错误代码。此外，还可以通过输出 xmlDoc.parseError.reason 和 xmlDoc.parseError.line 准确地定位错误的原因和错误的行数。代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<!--引入外部 DTD 文件-->
<html>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Insert title here</title>
        <style type="text/css">
        </style>
</head>
<script type="text/javascript">
<!--声明 script 类型-->
function validateXML(filename){
        var txt="";
        if (window.ActiveXObject){
                document.getElementById("showError").innerText ="";
                var xmlDoc = new ActiveXObject("MSXML2.DOMDocument.4.0");
                xmlDoc.async = false;
                xmlDoc.validateOnParse= true;
                xmlDoc.load(filename);
                if(xmlDoc.parseError.errorCode!=0){
                txt="Error Code: " + xmlDoc.parseError.errorCode + "\n";
                txt=txt+"Error Reason: " + xmlDoc.parseError.reason ;
                txt=txt+"Error Line: " + xmlDoc.parseError.line;
        }else{
                txt="没有错误";
              }
                document.getElementById("showError").innerText = txt;
        }else{
                alert("此浏览器不支持验证!");
        }
}
</script>
<body>
<center>
<h2>通过 XSD 验证 XML 文件</h2>
<!--创建 XML 文件标题-->
<p>
<input id="xmlfile" type="text" />
<input type="button" name="submit" value="验  证" onClick="validateXML(document.getElementById('xmlfile').value);" />
```

```
<!--添加 button 按钮-->
</p>
</center>
<div id="showError"></div>
</body>
</html>
```

（1）设计一个 HTML 文档，添加一个 text 标签接收需要验证的 XML。代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<!--引入外部的 DTD 文件-->
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
<style type="text/css">
<!--
#showError {
        color: #F00;
        font-size: 16px;
}
-->
</style></head>
<script type="text/javascript">
<!--声明 script 的类型-->
function validateXML(filename){
        var txt="";
        if (window.ActiveXObject){
                document.getElementById("showError").innerText ="";
                var xmlDoc = new ActiveXObject("MSXML2.DOMDocument.4.0");
                xmlDoc.async = false;
                xmlDoc.validateOnParse= true;
                xmlDoc.load(filename);
                if(xmlDoc.parseError.errorCode!=0){
                txt="Error Code: " + xmlDoc.parseError.errorCode + "\n";
                txt=txt+"Error Reason: " + xmlDoc.parseError.reason ;
                txt=txt+"Error Line: " + xmlDoc.parseError.line;
              }else{
                txt="没有错误";
              }
                document.getElementById("showError").innerText = txt;
        }else{
                        alert("此浏览器不支持验证!");
        }
}
</script>
<body>
<center>
<!--XML 文件居中显示-->
<h2>通过 XSD 验证 XML 文件</h2>
<!--创建 XML 标题-->
<p>
<input id="xmlfile" type="text" />
</p>
</center>
</body>
</html>
```

（2）在 HTML 中添加一个 button 按钮，在该按钮上添加 onClick()方法。这样每次单击该按钮时，都将触发验证 XML 文档的 JavaScript 方法。代码如下：

```
<input type="button" name="submit" value="验  证" onClick="validateXML(document.getElementById('xmlfile').value);" />
<!--添加 button 按钮，并声明 onClick()方法-->
```

（3）在 HTML 中添加一个 id 为 showError 的 div 标签。代码如下：

```
<div id="showError"></div>
```

（4）完成验证 DTD 的方法，在方法中把错误的信息输出到 showError 的位置上。

秘笈心法

心法领悟 013：XML 文档的验证方式。

本例使用 MSXML2.DOMDocument.4.0 对文档进行验证，这需要操作系统和浏览器的支持。有些 XML 编辑器也可以做到这一点，例如 Altova XMLSpy、XML Copy Editor 等，读者可以根据自己的爱好选择使用。

实例 014 XSD 文档根元素的引用

光盘位置：光盘\MR\01\014

初级

实用指数：★★★★

实例说明

XSD 本身也是一种 XML 文档，遵守 XML 的语法规范。在定义 XSD 时，诸如大小写等问题一定要注意，一些特殊标识符尽量少用。例如，定义一个空的 XSD 文档。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
        targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
        elementFormDefault="qualified">
</xs:schema>
```

关键技术

在这个 XSD 文档中使用了 xs 作为前缀，声明 xs 的命名空间为 http://www.w3.org/2001/XMLSchema。在文档中使用的元素和数据类型都需要以 xs 开头，如 xs:schema。具体代码如下：

```
xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

显示在 xs 下所定义的元素都来自 http://www.mingrisoft.com 的命名空间。具体代码如下：

```
targetNamespace="http://www.mingrisoft.com"
```

默认的命名空间来自 http://www.mingrisoft.com。具体代码如下：

```
xmlns="http://www.mingrisoft.com"
```

使用刚刚定义的命名空间限定 XML 实例文档使用的所有元素和类型。具体代码如下：

```
elementFormDefault="qualified"
```

设计过程

（1）建立一个 XSD 文档，为其定义 XML 版本和编码格式。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
```

（2）创建 XSD 的根节点 xs:schema。代码如下：

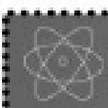
```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<--创建 XSD 的根节点-->
</xs:schema>
```

（3）在根节点开始的元素上定义文档的命名空间。代码如下：

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的名称空间-->
        targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com">
</xs:schema>
```

心法领悟 014：XSD 命名空间的限定。

实例声明 XSD 的命名空间时，如果 elementFormDefault="unqualified"，表示使用刚刚定义的命名空间限定 XML 实例文档使用的全局元素和类型，对局部元素不作限定。



实例 015　在 XSD 中设定元素的出现顺序

光盘位置：光盘\MR\01\015

初级

实用指数：★★★★

实例说明

xs:sequence 要求 XML 中的元素以指定顺序排列，在 xs:sequence 内声明的 xs:element 顺序就是在 XML 文档中使用的顺序。如果在 XML 中没有按照 xs:sequence 规定的顺序使用，验证就会出现如图 1.14 所示的错误。



图 1.14　在 XSD 中设定元素的出现顺序

关键技术

xs:complexType 可以声明一个复杂类型。在 XSD 中使用复杂类型约束 XML 的元素，需要联合使用 xs:complexType 与其他元素。使用 xs:complexType 定义一个复杂类型，在 xs:complexType 内部再定义一个 xs:sequence，它们共同对 XML 元素或属性进行限定。例如：

```
<xs:complexType>
<!--声明使用复杂类型-->
    <xs:sequence>
    <!--限定元素的排列顺序-->
        <xs:element name="name" type="xs:string" />
        <xs:element name="publisher" type="xs:string" />
        <xs:element name="company" type="xs:string" />
        <xs:element name="author" type="xs:string" />
        <xs:element name="ISBN" type="xs:string" />
        <xs:element name="price" type="xs:string" />
        <xs:element name="url" type="xs:string" />
    </xs:sequence>
</xs:complexType>
```

（1）建立 XSD 文档，声明文档根元素和命名空间。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
    targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
    elementFormDefault="qualified">
</xs:schema>
```

（2）使用 xs:element 定义 XML 元素的根节点 book，在 element 内部定义一个复杂类型 xs:complexType。代码如下：

```
<xs:element name="book">
<!--定义 XML 元素根节点-->
    <xs:complexType>
    </xs:complexType>
</xs:element>
```

（3）在 xs:complextpType 内部添加 xs:sequence 声明一个有顺序的元素定义，然后在 xs:sequence 内部使用 xs:element 声明 book 的子元素 name、publisher、company、author、ISBN、price、url，并且定义这几个元素为 xs:string 类型。代码如下：

```
<xs:sequence>
```

```
<!--声明元素的排列顺序-->
            <xs:element name="name" type="xs:string" />
            <xs:element name="publisher" type="xs:string" />
            <xs:element name="company" type="xs:string" />
            <xs:element name="author" type="xs:string" />
            <xs:element name="ISBN" type="xs:string" />
            <xs:element name="price" type="xs:string" />
            <xs:element name="url" type="xs:string" />
</xs:sequence>
```

（4）建立相应的 XML 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.mingrisoft.com simple_demo.xsd">
      <!--声明 XML Schema 的存储地址-->
      <name>《C#从入门到精通(第 2 版)》</name>
      <publisher>清华大学出版社</publisher>
      <company>明日科技</company>
      <author>王小科</author>
      <ISBN>9787302226628</ISBN>
      <price>69.80</price>
      <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 015：在 xs:element 中使用 xs:complexType 定义的复杂类型。

在 xs:element 中可以使用 xs:complexType 定义的一个复杂类型。有时需要定义很多一样的复杂类型要写很多次同样的定义，这时可以用 xs:complexType 提取出来，声明一个公共的复杂类型供其他的 xs:element 使用。将 xs:complexType 定义提取出来后，要为它添加一个属性 name。name 在文档中必须是唯一的，这样才能保证在引用时不会有冲突。例如：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
      targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
      elementFormDefault="qualified">
      <xs:element name="book" type="bookType" />
      <xs:complexType name="bookType">
      <!--声明复杂类型-->
            <xs:sequence>
                  <xs:element name="name" type="xs:string" />
                  <xs:element name="publisher" type="xs:string" />
                  <xs:element name="company" type="xs:string" />
                  <xs:element name="author" type="xs:string" />
                  <xs:element name="ISBN" type="xs:string" />
                  <xs:element name="price" type="xs:string" />
                  <xs:element name="url" type="xs:string" />
            </xs:sequence>
      </xs:complexType>
</xs:schema>
```

实例 016 在 XSD 中使用扩展数据类型

光盘位置：光盘\MR\01\016

高级

实用指数：★★★★☆

实例说明

在一本图书的 XML 中，price 表示图书的价格。在某些特殊的情况下，图书的价格可能会有两个，一个是原价，另一个是打折以后的价格。那么在 price 中就会有 orginal 和 dicount 两个子元素，分别记录原价和打折以后的价格。本例将在 XSD 中使用扩展数据类型，运行结果如图 1.15 所示。

关键技术

通过 xs:complexContent 和 xs:extension 可以使用外部定义的元素和属性。例如，在 XSD 中定义 price 时有两个价格，orginal 表示原价，可以直接定义在 xs:extension 内部；discount 表示折扣价格，一般都是后期根据库存等情况制定的，所以将其定义在 price 元素外部，在使用时需要用 xs:extension 的 base 属性来引用。

在 xs:extension 内部定义元素 original，同时设置 xs:extension 的属性 base="discountPrice"引用外部定义的 discount 元素，这样既保证了原有的 original 元素，又引用了 discount 元素。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
        targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
        elementFormDefault="qualified">
        <xs:element name="book">
                <xs:complexType>
                <!--声明复杂类型-->
                        <xs:sequence>
                        <!--声明元素的排列顺序-->
                                <xs:element name="name" type="xs:string" />
                                <xs:element name="publisher" type="xs:string" />
                                <xs:element name="company" type="xs:string" />
                                <xs:element name="author" type="xs:string" />
                                <xs:element name="ISBN" type="xs:string" />
                                <xs:element name="price">
                                        <xs:complexType>
                                                <xs:complexContent>
                                                        <xs:extension base="discountPrice">
                                                                <xs:sequence>
                                                                        <xs:element name="original" type="xs:double" />
                                                                </xs:sequence>
                                                                <xs:attribute name="RMB" type="xs:string" use="required" />
                                                        </xs:extension>
                                                </xs:complexContent>
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="url" type="xs:string" />
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        <xs:complexType name="discountPrice">
        <!--声明复杂类型-->
                <xs:sequence>
                        <xs:element name="discount" type="xs:double" />
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个 book 根节点及其子元素（name、publisher、company、author、ISBN 和 url），为各个子元素添加 xs:string 类型限定。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
        targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
        elementFormDefault="qualified">
        <xs:element name="book">
                <xs:complexType>
                        <xs:sequence>
                        <!--声明元素的排列顺序-->
                                <xs:element name="name" type="xs:string" />
```

```
                        <xs:element name="publisher" type="xs:string" />
                        <xs:element name="company" type="xs:string" />
                        <xs:element name="author" type="xs:string" />
                        <xs:element name="ISBN" type="xs:string" />
                        <xs:element name="url" type="xs:string" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
</xs:schema>
```

（2）为 price 声明一个复杂类型，在 xs:complexType 中扩展 xs:complexContent、xs:extension 声明，设置 xs:extension 属性 base="discountPrice"。代码如下：

```
<xs:element name="price">
                <xs:complexType>
                <!--声明复杂类型-->
                    <xs:complexContent>
                        <xs:extension base="discountPrice">
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
</xs:element>
```

（3）在 xs:extension 的基础上扩展 xs:attribute 属性，同时定义 name="RMB"、type="xs:string"和 use="required"。代码如下：

```
<xs:sequence>
                <!--声明元素排列方式-->
                <xs:element name="original" type="xs:double" />
                </xs:sequence>
<xs:attribute name="RMB" type="xs:string" use="required" />
```

（4）在声明 book 元素的外部定义一个 xs:complexType，扩展 xs:complexType 定义 discount 元素，设置 xs:complexType 的 name="discountPrice"。代码如下：

```
<xs:complexType name="discountPrice">
        <xs:sequence>
                <xs:element name="discount" type="xs:double" />
        </xs:sequence>
</xs:complexType>
```

■ 秘笈心法

心法领悟 016：xs:complexContent 和 xs:extension 的使用注意点。

使用 xs:complexContent 和 xs:extension 定义元素 original 的同时，也会引用一个复杂类型，其中定义了 discount 元素。这两个元素分别定义在不同的 xs:sequnce 内部，但是在 XML 中使用时要先使用 xs:extension 通过 base 引用的元素，即 discount，再使用 xs:extension 内部定义的元素，否则在验证时会出现如图 1.16 所示的错误。

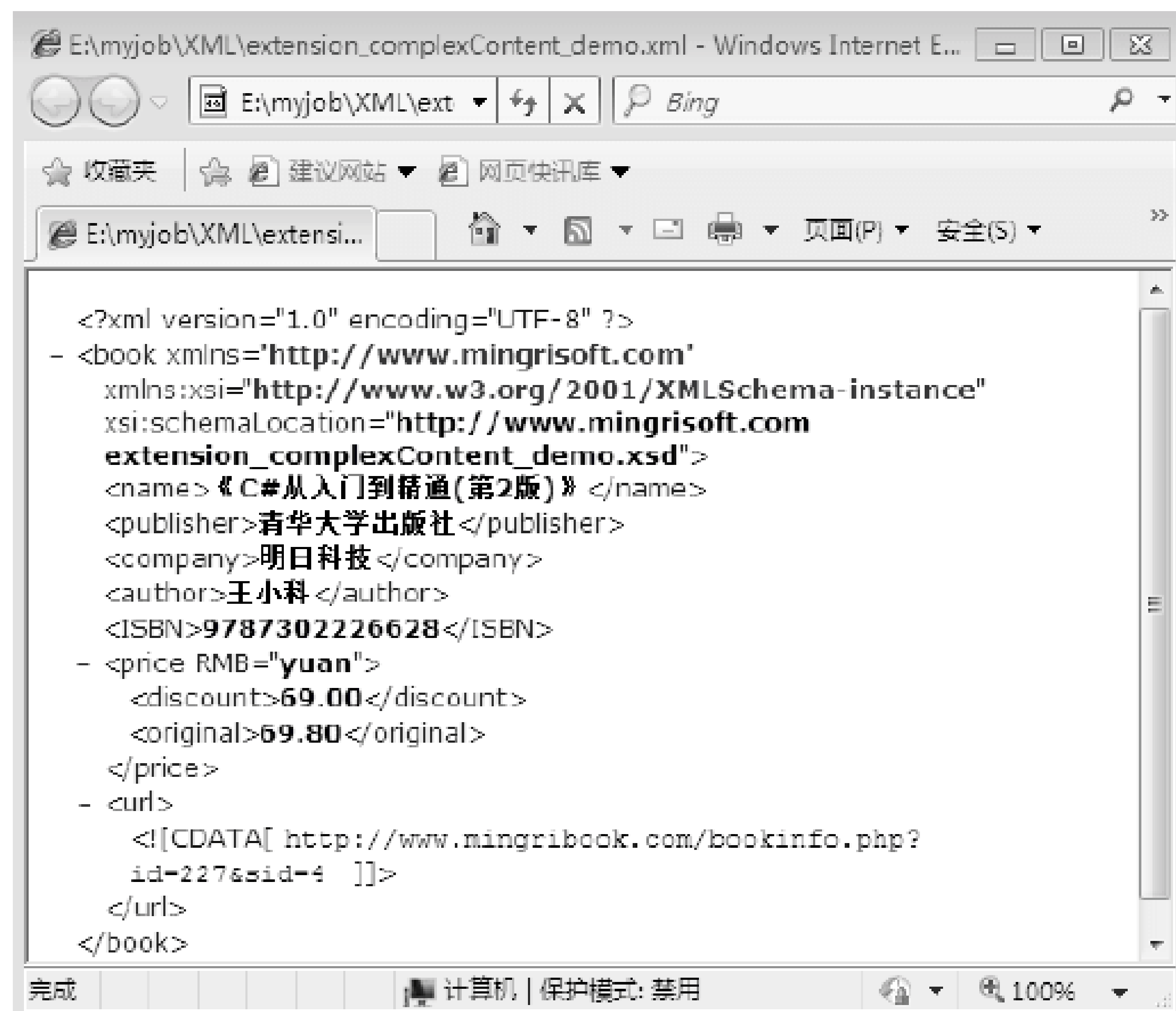


图 1.15　在 XSD 中使用扩展数据类型



图 1.16　通过 xs:extension 引入的元素顺序使用错误

实例 017　**在 XSD 中使用元素的条理化**　高级

光盘位置：光盘\MR\01\017　实用指数：★★★★★

实例说明

在定义 XML 格式时，会声明很多重复的内容。此时最好的方法就是把重复的内容提取出来，以后每次声明时只需调用提取出来的声明内容即可，如图 1.17 所示。但是不管 XML 是怎样声明的，最后使用时都是一样的，不会有太大差别。

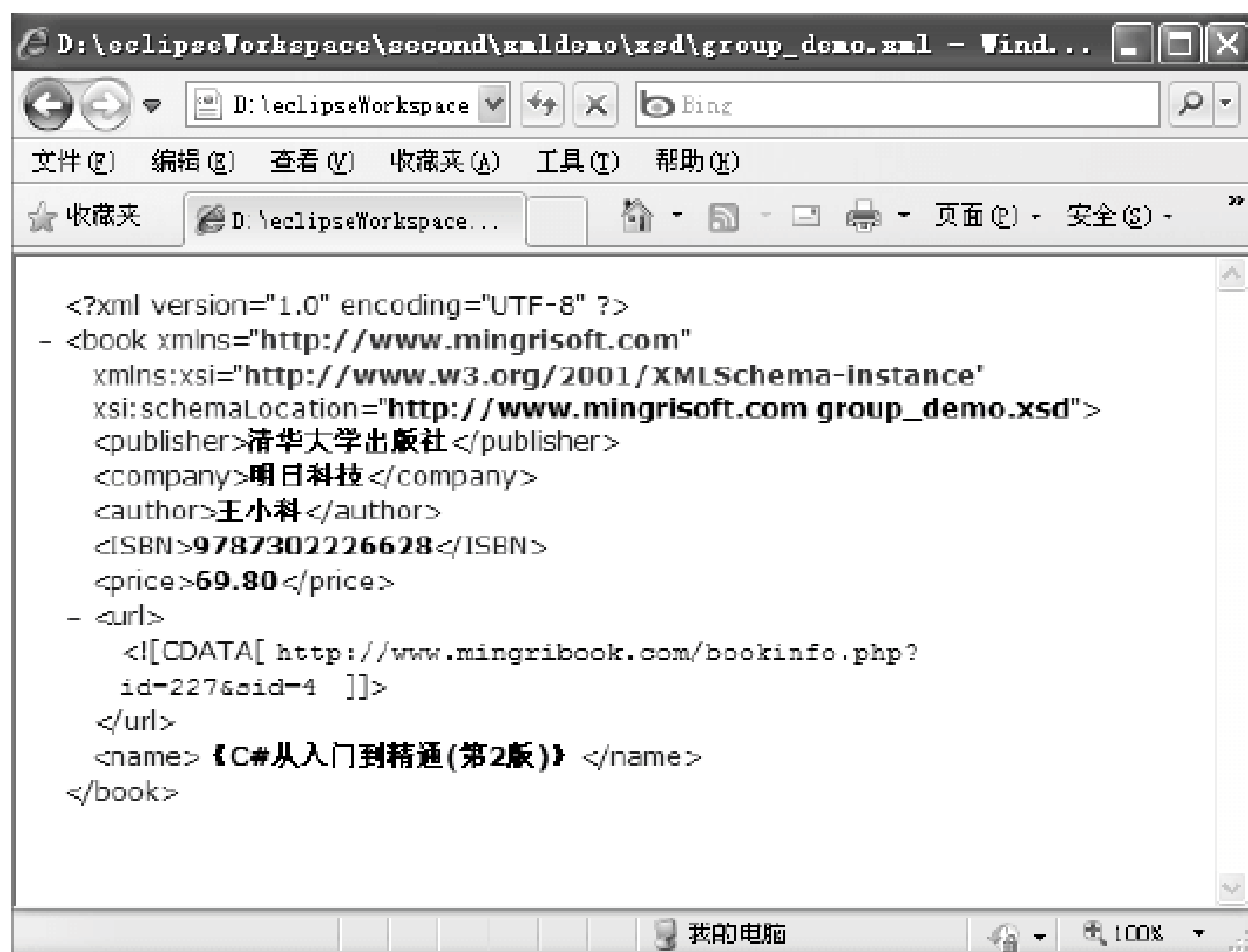


图 1.17　在 XSD 中使用元素的条理化

在 xs:group 内部可以声明 xs:choice、xs:sequence 和 xs:all 元素，而在 xs:choice、xs:sequence 或者 xs:all 内部又都可以定义多个 xs:element 元素，并为每个 xs:element 声明各自的名称和类型，这样就形成了一个 xs:element 组。接下来声明 xs:group 的 name="otherType"，并为该组命名。在文档的其他地方需要使用时，定义 xs:group 并且使用 ref="otherType"即可引用这个组。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
    targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
    elementFormDefault="qualified">
    <xs:element name="book">
        <xs:complexType>
            <xs:sequence>
            <!--声明元素的排列顺序-->
                <xs:group ref="otherType"></xs:group>
                <xs:element name="name" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:group name="otherType">
        <xs:sequence>
        <!--声明元素的排列顺序-->
            <xs:element name="publisher" type="xs:string" />
            <xs:element name="company" type="xs:string" />
            <xs:element name="author" type="xs:string" />
            <xs:element name="ISBN" type="xs:string" />
            <xs:element name="price" type="xs:double" />
            <xs:element name="url" type="xs:string" />
        </xs:sequence>
```

```
        </xs:group>
</xs:schema>
```

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。定义一个 xs:element 的 name="book"，都定义一个根节点。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
        targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
        elementFormDefault="qualified">
        <xs:element name="book">
        </xs:element>
</xs:schema>
```

（2）在根节点下声明一个复杂类型 xs:complexType，使用 xs:sequence 扩展 xs:complexType。代码如下：

```
<xs:complexType>
<!--声明复杂类型-->
                <xs:sequence>
                </xs:sequence>
</xs:complexType>
```

（3）在 xs:sequence 下定义一个 xs:element 元素，设置 name="name"，表示在根元素 book 下定义一个子节点，并为子节点设置 type="xs:string"。代码如下：

```
<xs:element name="name" type="xs:string" />
<!--声明子节点-->
```

（4）独立声明一个 xs:group，设置其属性 name="otherType"。扩展 xs:sequence 元素，在其下定义一组子元素 publisher、company、author、ISBN、price 和 url。代码如下：

```
<xs:group name="otherType">
<!--声明 xs:group 设置 xs:group 属性-->
        <xs:sequence>
        <!--声明元素的排列顺序-->
                <xs:element name="publisher" type="xs:string" />
                <xs:element name="company" type="xs:string" />
                <xs:element name="author" type="xs:string" />
                <xs:element name="ISBN" type="xs:string" />
                <xs:element name="price" type="xs:double" />
                <xs:element name="url" type="xs:string" />
        </xs:sequence>
</xs:group>
```

（5）在声明 book 根元素的里面、xs:element 的上面插入定义好的元素组 ref="otherType"。代码如下：

```
<xs:group ref="otherType"></xs:group>
<!--声明插入元素组-->
```

秘笈心法

心法领悟 017：xs:group 中的顺序限制。

在使用 xs:group 时，把 xs:group 引用放在了 xs:element 的上面，由于它们都被包含在 xs:sequence 内部，所以说明了 xs:group 元素在 XML 文档中是有顺序限制的。同时在 xs:group 内部也使用了 xs:sequence，说明 xs:group 内部也是有顺序限制的。把这两部分拼在一起，XML 元素就有了顺序排列。



实例 018	XSD 中的多属性打包 光盘位置：光盘\MR\01\018	高级 实用指数：★★★★★

实例说明

定义 XSD 时，除了元素外还有属性；在使用 XML 时属性也占有相当大的比例，由此可见属性十分重要。

例如，定义一本图书的 XML 元素有图书名称、出版社、作者、价格等。以作者为例，可以将其联系方式作为一种属性，如 qq、msn、tel，如图 1.18 所示。

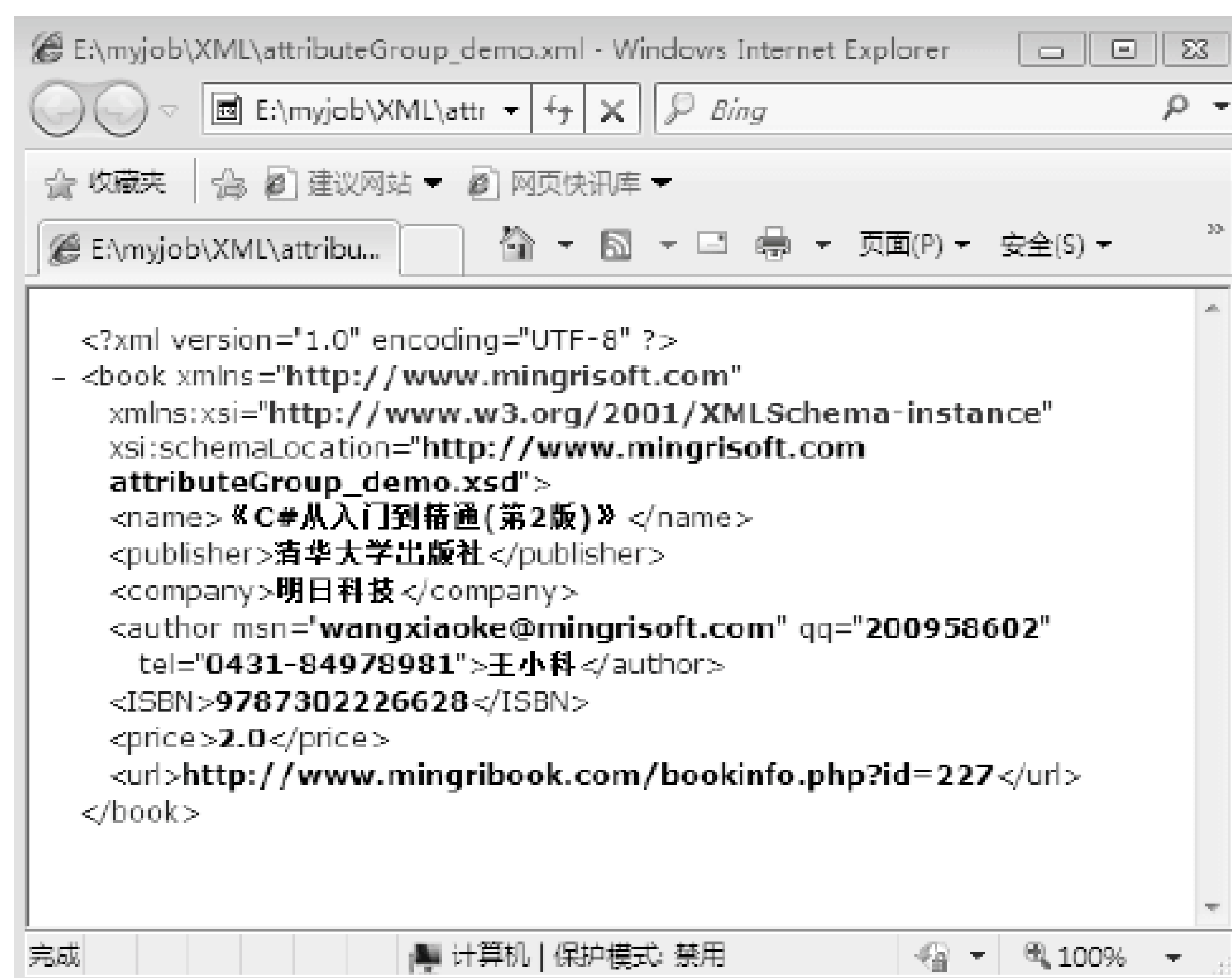


图 1.18　XSD 中的多属性打包

关键技术

使用 xs:attributeGroup 定义一个属性组，设置 name="att"（为组命名），然后在 xs:attributeGroup 内部定义属性组的成员，成员数量没有限制。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
    targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
    elementFormDefault="qualified">
    <xs:element name="book">
        <xs:complexType>
        <!--声明复杂类型-->
            <xs:sequence>
            <!--声明元素的排列顺序-->
                <xs:element name="name" type="xs:string" />
                <xs:element name="publisher" type="xs:string" />
                <xs:element name="company" type="xs:string" />
                <xs:element name="author">
                    <xs:complexType>
                        <xs:simpleContent>
                            <xs:extension base="xs:string">
                                <xs:attributeGroup ref="att"></xs:attributeGroup>
                            </xs:extension>
                        </xs:simpleContent>
                    </xs:complexType>
                </xs:element>
                <xs:element name="ISBN" type="xs:string" />
                <xs:element name="price" type="xs:double" />
                <xs:element name="url" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:attributeGroup name="att">
    <!--定义一个属性组-->
        <xs:attribute name="tel" type="xs:string" use="required"></xs:attribute>
        <xs:attribute name="qq" type="xs:integer"></xs:attribute>
        <xs:attribute name="msn" type="xs:string"></xs:attribute>
    </xs:attributeGroup>
</xs:schema>
```

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个根节点 book，添加复杂类型 xs:complexType

以及 book 下的子元素（name、publisher、company、ISBN、price 和 url），为各个子元素添加 xs:string 类型限定。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
        targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
        elementFormDefault="qualified">
        <xs:element name="book">
                <xs:complexType>
                        <xs:sequence>
                        <!--声明元素的排列顺序-->
                                <xs:element name="name" type="xs:string" />
                                <xs:element name="publisher" type="xs:string" />
                                <xs:element name="company" type="xs:string" />
                                <xs:element name="ISBN" type="xs:string" />
                                <xs:element name="price" type="xs:double" />
                                <xs:element name="url" type="xs:string" />
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
</xs:schema>
```

（2）在 company 下面添加一个元素 author，然后在 xs:simpleType 上扩展 xs:simpleType 和 xs:extension 元素，在 xs:extension 上为 author 内容声明属性 xs:string。代码如下：

```
<xs:element name="author">
                <xs:complexType>
                <!--声明复杂类型-->
                        <xs:simpleContent>
                        </xs:simpleContent>
                </xs:complexType>
</xs:element>
```

（3）在 xs:schema 内部和 book 根节点平级的地方，声明一个 xs:attributeGroup，设置其 name="att"；在 xs:attributeGroup 内部声明 3 个属性，分别为 tel、qq 和 msn，形成一个具有 3 个属性的属性组。代码如下：

```
<xs:attributeGroup name="att">
<!--声明一个属性组-->
        <xs:attribute name="tel" type="xs:string" use="required"></xs:attribute>
        <xs:attribute name="qq" type="xs:integer"></xs:attribute>
        <xs:attribute name="msn" type="xs:string"></xs:attribute>
</xs:attributeGroup>
```

（4）在声明 author 的位置，扩展 xs:extension 元素，在其内部声明一个 xs:attributeGroup 引用，引用刚才声明的属性组。代码如下：

```
<xs:extension base="xs:string">
                <xs:attributeGroup ref="att"></xs:attributeGroup>
                <!--引入属性组-->
</xs:extension>
```

秘笈心法

心法领悟 018：author 属性组中的属性。

在本实例中为 author 添加了一个属性组，内部有 3 个属性 tel、qq、msn，这样是为了更好地说明 xs: attributeGroup 的使用方式，但一般不建议这样定义 XML，最好的方式是把 tel、qq 和 msn 定义成 author 的子元素。

实例 019 XSD 中对元素的限定

高级

光盘位置：光盘\MR\01\019

实用指数：★★★★★

实例说明

对于 XML 的元素内容，可以通过 xs:enumeration 将其定义成枚举值，也可以使用 xs:pattern 将其定义成一

定的样式，使其符合一定的规则，如果违反了这种规则，解析器将检测出错误，如图 1.19 所示。



图 1.19　XSD 中对元素的限定

■ 关键技术

xs:pattern 被用在 xs:restriction 内部，用来限定 XML 中元素使用固定的格式。格式的写法使用正则表达式来表示。例如，value="[0-9]"表示 0～9 之间的任意一个数字，那么 value="[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9]"就表示应该有 13 个数字，每个数字只能是 0～9，在实际应用中这符合 ISBN 的规则。

■

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个根节点 book 及其子元素（name、publisher、company、author、price 和 url），并为各子元素添加 xs:string 类型限定。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
	targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
	elementFormDefault="qualified">
	<xs:element name="book" type="bookType" />
	<xs:complexType name="bookType">
	<!--声明复杂元素 bookType-->
		<xs:sequence>
		<!--声明元素的排列顺序-->
			<xs:element name="name" type="xs:string" />
			<xs:element name="publisher" type="xs:string" />
			<xs:element name="company" type="xs:string" />
			<xs:element name="author" type="xs:string" />
			<xs:element name="price" type="xs:string" />
			<xs:element name="url" type="xs:string" />
		</xs:sequence>
	</xs:complexType>
</xs:schema>
```

（2）为 ISBN 声明一个简单类型，定义 xs:restriction 为 xs:integer。代码如下：

```
<xs:element name="ISBN">
	<xs:simpleType>
	<!--声明简单类型-->
		<xs:restriction base="xs:integer">
		</xs:restriction>
	</xs:simpleType>
</xs:element>
```

（3）在 xs:restriction 内定义一个 xs:parttern，设置 value="[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9]"。代码如下：

```
<xs:pattern value="[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]" />
<!--设置 value 的值-->
```

（4）在 XML 中使用 ISBN 时，元素内部必须填写 13 个连续的数字，每个数字在 0～9 之间任意取值即可。

秘笈心法

心法领悟 019：在 xs:pattern 中的正则表达式。

在 xs:pattern 中使用正则表达式，可以通过某种模式去匹配一类字符串的一个公式。使用正则表达式来表示字符串非常灵活，而且表达起来很简单，功能还很强大，有兴趣的读者可以自己查阅相关资料，在此不再赘述。

实例 020　在 XSD 中使用取值范围的限定

光盘位置：光盘\MR\01\020

高级

实用指数：★★★★★

实例说明

在定义 XSD 时，有时会先声明一个基本的元素对象，然后根据不同的需要对其声明进行扩展，即在原有的元素声明基础上，不需要改变原来元素的定义，对原来的定义进一步完善。这时可以使用 xs:complexContent 和 xs:restriction，使用的 XML 如图 1.20 所示。

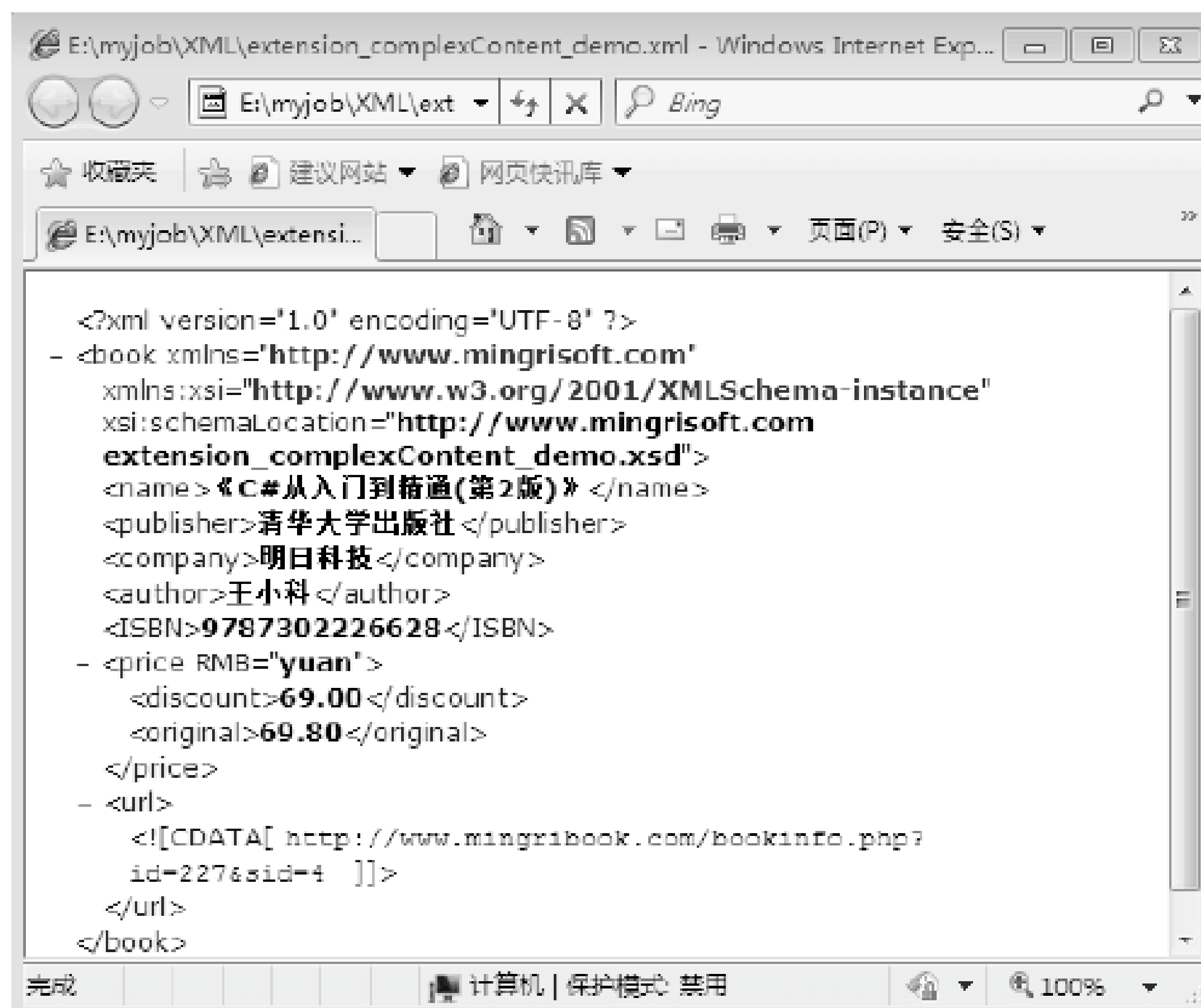


图 1.20　在 XSD 中使用取值范围的限定

在 XSD 中，为了扩展 baseBookType，要声明一个名为 bookType 的复杂元素，在 xs:complexType 内部扩展 xs:complexContent 和 xs:restriction，并设置 xs:restriction 的 base="baseBookType"。同时在 xs:restriction 内部对基本的元素声明进行扩展，设置元素 company 的属性 fixed="明日科技"，对原有的 company 定义进行扩展。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
        targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
        elementFormDefault="qualified">
        <xs:element name="books">
                <xs:complexType>
                        <xs:all>
                                <xs:element name="book" type="bookType" />
                        </xs:all>
                </xs:complexType>
        </xs:element>
        <xs:complexType name="baseBookType">
        <!--声明复杂元素 baseBookType-->
                <xs:sequence>
                <!--声明元素的排列顺序-->
```

```
                <xs:element name="name" type="xs:string" />
                <xs:element name="publisher" type="xs:string" />
                <xs:element name="company" type="xs:string" />
                <xs:element name="author" type="xs:string" />
                <xs:element name="ISBN" type="xs:string" />
                <xs:element name="price" type="xs:double" />
                <xs:element name="url" type="xs:string" />
            </xs:sequence>
    </xs:complexType>
    <xs:complexType name="bookType">
    <!--声明复杂元素 bookType-->
        <xs:complexContent>
            <xs:restriction base="baseBookType">
                <xs:sequence>
                <!--声明元素的排列顺序-->
                    <xs:element name="name" type="xs:string" />
                    <xs:element name="publisher" type="xs:string" />
                    <xs:element name="company" type="xs:string" fixed="明日科技" />
                    <xs:element name="author" type="xs:string" />
                    <xs:element name="ISBN" type="xs:string" />
                    <xs:element name="price" type="xs:double" />
                    <xs:element name="url" type="xs:string" />
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个根节点 books，在其下面定义子元素 book。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
    targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
    elementFormDefault="qualified">
    <xs:element name="books">
    <!--声明根节点 books-->
        <xs:complexType>
            <xs:all>
            </xs:all>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

（2）声明一个复杂类型 baseBookType 作为基本的元素声明，然后定义其子元素 name、publisher、company、author、ISBN、price 和 url，并为各个子元素添加 xs:string 类型限定，为 price 添加 xs:double 类型。代码如下：

```
<xs:complexType name="baseBookType">
<!--声明复杂元素 baseBookType-->
    <xs:sequence>
    <!--声明元素的排列顺序-->
        <xs:element name="name" type="xs:string" />
        <xs:element name="publisher" type="xs:string" />
        <xs:element name="company" type="xs:string" />
        <xs:element name="author" type="xs:string" />
        <xs:element name="ISBN" type="xs:string" />
        <xs:element name="price" type="xs:double" />
        <xs:element name="url" type="xs:string" />
    </xs:sequence>
</xs:complexType>
```

（3）声明另外一个复杂类型 bookType，扩展 xs:complexContent 和 xs:restriction 元素，并设置 xs:restriction 属性 base="baseBookType"，覆盖 baseBookType 的声明，然后添加元素 company 的属性 fixed="明日科技"，表示 company 内容已经被限定，只能是“明日科技”。代码如下：

```
<xs:complexType name="bookType">
```

```
<!--声明复杂类型-->
    <xs:complexContent>
        <xs:restriction base="baseBookType">
            <xs:sequence>
            <!--声明元素的排列顺序-->
                <xs:element name="name" type="xs:string" />
                <xs:element name="publisher" type="xs:string" />
                <xs:element name="company" type="xs:string" fixed="明日科技" />
                <xs:element name="author" type="xs:string" />
                <xs:element name="ISBN" type="xs:string" />
                <xs:element name="price" type="xs:double" />
                <xs:element name="url" type="xs:string" />
            </xs:sequence>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>
```

（4）在根元素的内部添加 xs:element 元素 book，设置类型为 bookType。代码如下：

```
<xs:element name="book" type="bookType" />
```

秘笈心法

心法领悟 020：xs:complexContent 和 xs:restriction 的扩展。

在使用 xs:complexContent 和 xs:restriction 对原有的基本元素声明进行扩展时，要把 xs:restriction 引用定义的基本声明进行重写覆盖，然后在此基础上进行扩展，否则不符合 xs:complexContent 和 xs:restriction 的使用规范。

实例 021 在 XSD 中声明元素属性

光盘位置：光盘\MR\01\021

高级

实用指数：★★★★★

实例说明

定义 XSD 时，除了元素外还有属性，在使用 XML 时属性也占有很大比重。例如，定义一本图书的 XML 元素有图书名称、出版社、作者、价格等。以作者为例，可以将其联系方式作为一种属性，如 qq、msn、tel，如图 1.21 所示。

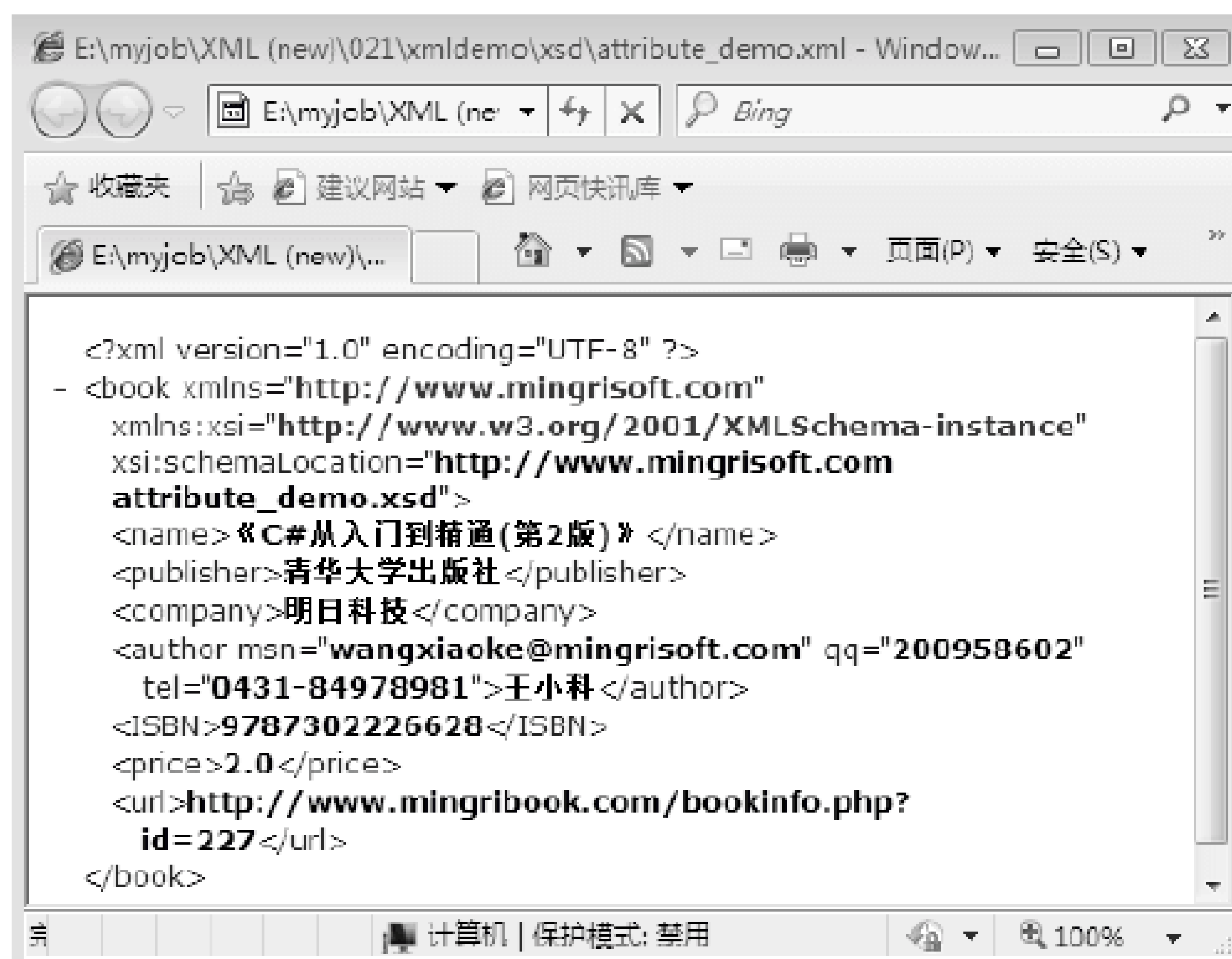


图 1.21　在 XSD 中声明元素属性

定义属性可以使用 xs:attribute，其中的 name 用于设置属性名称，type 用于设置属性类型。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
```

```
        elementFormDefault="qualified">
        <xs:element name="book">
        <!--声明根节点 book-->
                <xs:complexType>
                        <xs:sequence>
                        <!--声明元素的排列顺序-->
                                <xs:element name="name" type="xs:string" />
                                <xs:element name="publisher" type="xs:string" />
                                <xs:element name="company" type="xs:string" />
                                <xs:element name="author">
                                        <xs:complexType>
                                        <!--声明复杂类型-->
                                                <xs:simpleContent>
                                                        <!--声明简单类型-->
                                                        <xs:extension base="xs:string">
                                                                <xs:attributeGroup ref="att"></xs:attributeGroup>
                                                                <!--声明引入一个属性组-->
                                                        </xs:extension>
                                                </xs:simpleContent>
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="ISBN" type="xs:string" />
                                <xs:element name="price" type="xs:double" />
                                <xs:element name="url" type="xs:string" />
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        <xs:attributeGroup name="att">
        <!--声明一个属性组-->
                <xs:attribute name="tel" type="xs:string" use="required"></xs:attribute>
                <xs:attribute name="qq" type="xs:integer"></xs:attribute>
                <xs:attribute name="msn" type="xs:string"></xs:attribute>
        </xs:attributeGroup>
</xs:schema>
```

（1）建立 XSD 文档，声明文档根元素和命名空间；然后声明一个 book 根节点，添加复杂类型 xs:complexType 以及 book 下的子元素（name、publisher、company、ISBN、price 和 url），并为各个子元素添加 xs:string 类型限定。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
        targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
        elementFormDefault="qualified">
        <xs:element name="book">
                <xs:complexType>
                <!--声明复杂类型-->
                        <xs:sequence>
                        <!--声明元素的排列顺序-->
                                <xs:element name="name" type="xs:string" />
                                <xs:element name="publisher" type="xs:string" />
                                <xs:element name="company" type="xs:string" />
                                <xs:element name="ISBN" type="xs:string" />
                                <xs:element name="price" type="xs:double" />
                                <xs:element name="url" type="xs:string" />
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
</xs:schema>
```

（2）在 company 下面添加一个元素 author，并为其声明属性，即在 xs:simpleType 上扩展 xs:simpleType 和 xs:extension 元素，在 xs:extension 上为 author 内容声明属性 xs:string。代码如下：

```
<xs:element name="author">
<!--声明元素及属性-->
```

```
        <xs:complexType>
                <!--声明复杂类型-->
                <xs:simpleContent>
                </xs:simpleContent>
        </xs:complexType>
</xs:element>
```

（3）在 author 内部声明 3 个属性，分别为 tel、qq 和 msn，形成一个具有 3 个属性的属性组。代码如下：

```
<xs:attribute name="tel" type="xs:string" use="required"></xs:attribute>
<!--声明元素形成属性组-->
        <xs:attribute name="qq" type="xs:integer"></xs:attribute>
        <xs:attribute name="msn" type="xs:string"></xs:attribute>
```

秘笈心法

心法领悟 021：author 的子元素的声明。

在本实例中使用 xs:attribute 声明 3 个属性 tel、qq、msn，作为 author 的子元素。

实例 022 在 XSD 中对字符进行限制

光盘位置：光盘\MR\01\022

高级

实用指数：★★★★★

实例说明

在 XML 中，可以使用各种样式的字符串表达不同的含义。为了更好地支持 XML，XSD 定义了很多字符串类型来满足不同需要。在本实例中使用 XSD 定义了不同的字符串类型来描述一本图书，并通过建立 XML 来使用这些字符串类型，如图 1.22 所示。



图 1.22　在 XSD 中对字符进行限制

在 XSD 中定义的 name 的 type="xs:token"，限制了 name 元素中不能包含换行符、回车或制表符、开头或结尾空格或者多个连续空格的字符串，表示在使用 XML 时图书的名称不应该含有这些符号。

定义 author 中的 type="xs:Name"，说明 author 元素是一个只能包含数字、字母、下划线、冒号及其他名称字符的 token 类型，而且 author 的第一个字符还不能是数字。

定义 publisher 中的 type="xs:NCName"，说明 publisher 元素在使用时必须是一个不包含冒号的 xs:Name 类型。

定义 company 中的 type="xs:QName"，说明 company 元素在 XML 使用时可以由前缀名和局部名组成，前缀名和局部名都必须是 xs:NCName，中间使用冒号分隔开。

定义代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
        targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
        elementFormDefault="qualified">
        <xs:element name="book">
                <xs:complexType>
                <!--声明复杂类型-->
                        <xs:sequence>
                        <!--声明元素的排列顺序-->
                                <xs:element name="name" type="xs:token" />
                                <xs:element name="publisher" type="xs:NCName" />
                                <xs:element name="company" type="xs:QName" />
```

```
                <xs:element name="author" type="xs:Name" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间；然后声明一个 book 根节点，添加复杂类型 xs:complexType。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的名称空间-->
    targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
    elementFormDefault="qualified">
    <xs:element name="book">
        <xs:complexType>
        <!--声明复杂类型-->
            <xs:sequence>
            <!--声明元素的排列顺序-->
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

（2）在根元素 book 内部声明子元素 name、publisher、company 和 author，分别定义其数据类型为 xs:token、xs:NCName、xs:QName 和 xs:Name。代码如下：

```
<xs:element name="name" type="xs:token" />
<!--声明子元素，定义数据类型-->
        <xs:element name="publisher" type="xs:NCName" />
        <xs:element name="company" type="xs:QName" />
<xs:element name="author" type="xs:Name" />
```

（3）在 XML 中分别根据这几种不同的类型定义使用不同的元素。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<bo:book xmlns:bo="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.mingrisoft.com string_demo.xsd">
    <!--声明命名空间的存储地址-->
    <bo:name>《C#从入门到精通(第 2 版)》</bo:name>
    <bo:publisher>清华大学出版社</bo:publisher>
    <bo:company>bo:明日科技有限责任公司</bo:company>
    <bo:author>明日科技:王小科</bo:author>
</bo:book>
```

秘笈心法

心法领悟 022：xs:string 的子集。

本实例中使用的字符串定义符都是 xs:string 的子集，xs:string 表示所有的字符串都适用，字符串的内容可以是回车符、空格符、制表符等。xs:string 还有一个子集，即 xs:normalizedString。xs:normalizedString 是一个规范化子集，表示 xs:string 中除了回车符、换行符、制表符以外的所有字符串的一个类型。



实例 023	在 XSD 中对数值进行限制 光盘位置：光盘\MR\01\023	高级 实用指数：★★★★★

数值的类型在 XSD 中非常丰富，无论是数字正负、数字位数、数字的使用区间，还是数值的精确位置上都有很多类型可供选择。在本实例中使用 XSD 定义不同的数字类型，描述一本图书的 ISBN、price 和 pageNum，

如图 1.23 所示。

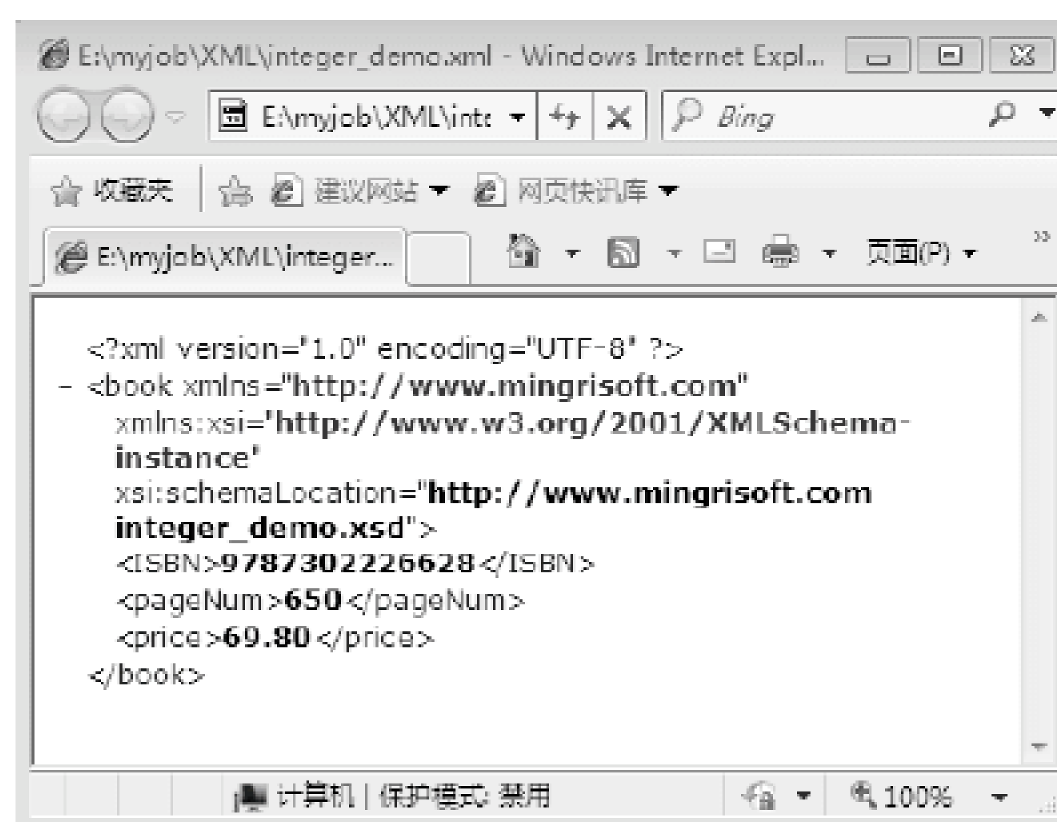


图 1.23　在 XSD 中对数值进行限制

关键技术

在 XSD 中定义 ISBN 的 type="xs:positiveInteger"，限制 ISBN 元素中的数据必须是正整数，数据内容不能包含 0。定义 pageNum 的 type="xs: nonNegativeInteger"，表示 pageNum 元素中的数据为非负值，数据内容可以包含 0。在定义 price 时，type="xs:double"，表示设置精度比较高的数据类型。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
        targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
        elementFormDefault="qualified">
        <xs:element name="book">
        <!--声明根节点 book-->
                <xs:complexType>
                <!--声明复杂类型-->
                        <xs:sequence>
                        <!--声明元素的排列顺序-->
                                <xs:element name="ISBN" type="xs:positiveInteger" />
                                <xs:element name="pageNum" type="xs:nonNegativeInteger" />
                                <xs:element name="price" type="xs:double" />
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
</xs:schema>
```

（1）建立 XSD 文档，声明文档根元素和命名空间；然后声明一个 book 根节点，声明复杂类型 xs:complexType。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!--声明文档的命名空间-->
        targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
        elementFormDefault="qualified">
        <xs:element name="book">
        <!--声明根节点 book-->
                <xs:complexType>
                        <xs:sequence>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
</xs:schema>
```

（2）在根元素 book 中声明子元素 ISBN、pageNum 和 price，分别为其定义 xs: positiveInteger、xs: nonNegativeInteger 和 xs: double 类型。代码如下：

```
<xs:element name="ISBN" type="xs:positiveInteger" />
```

```
<!--声明子元素，定义数据类型-->
            <xs:element name="pageNum" type="xs:nonNegativeInteger" />
            <xs:element name="price" type="xs:double" />
```

（3）在 XML 中分别根据这几种不同的类型定义使用不同的元素内容。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.mingrisoft.com integer_demo.xsd">
      <!--声明命名空间的存储地址-->
      <ISBN>9787302226628</ISBN>
      <pageNum>650</pageNum>
      <price>69.80</price>
</book>
```

秘笈心法

心法领悟 023：XSD 中数据类型扩展。

在本实例中使用的数据类型只是一小部分比较有代表性的，其实在 XSD 中还有很多数据类型值得学习，如 xs: byte 表示具有正负值的 8 位数，xs:int 表示有正负值的 32 位数，xs:long 表示有正负值的 64 位数等。

1.3　XML 解析

实例 024　使用 DOM 组件从文件中读取 XML

光盘位置：光盘\MR\01\024

初级

实用指数：★★★★

使用 DOM 解析 XML，首先要读取 XML 文件。一般对文件的操作都是使用 File 类，把文件的访问路径通过 File 的构造方法传递进去得到一个 File 的对象，然后对这个 File 对象进行读取。本例要读取的 XML 文件如图 1.24 所示。

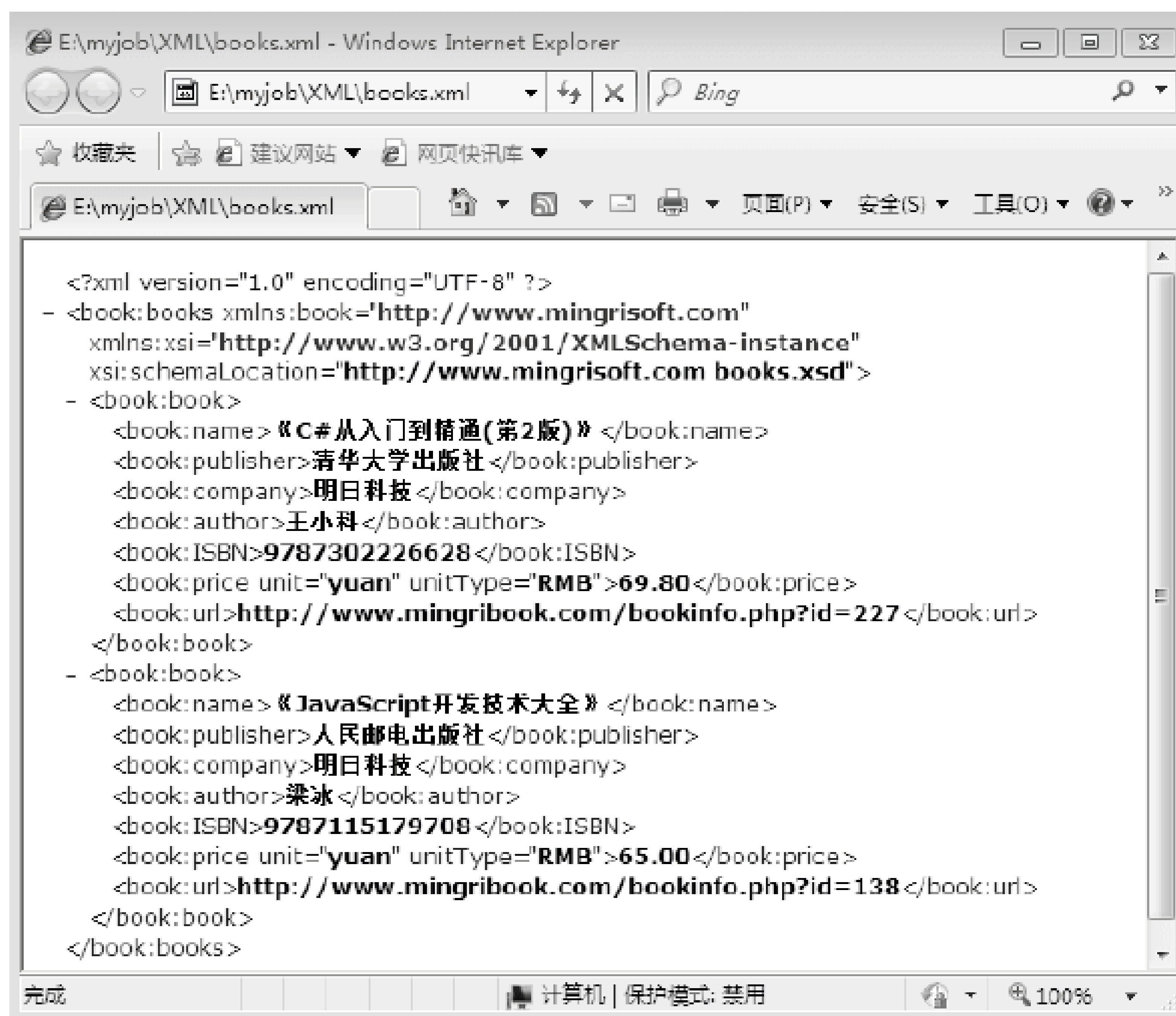


图 1.24　使用 DOM 组件从文件中读取 XML

关键技术

使用 DOM 解析 XML 文件前，先通过 DocumentBuilderFactory 的 newInstance()方法创建一个

DocumentBuilderFactory 的实例 documentBuilderFactory，通过 documentBuilderFactory 可以获取 DocumentBuilder 的实例 dombuilder，然后使用 dombuilder 的 parse()方法解析 File 对象。代码如下：

```
DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory
                .newInstance();                    //创建 DocumentBuilderFactory 的实例 documentBuilderFactory
DocumentBuilder dombuilder = documentBuilderFactory
                .newDocumentBuilder()              //获取到 DocumentBuilder 的实例 dombuilder
File file = new File(path);                        //获取文件
dombuilder.parse(file);                            //解析文件
```

设计过程

（1）分别创建 DocumentBuilderFactory 的实例 documentBuilderFactory 和 File 的实例 file。设置 File 实例时需要用到 XML 文档路径 path，path 可以通过 parseReadFile()方法的参数传递进来。代码如下：

```
public void parseReadFile(String path) throws ParserConfigurationException,SAXException, IOException {
        //创建 DocumentBuilderFactory 对象
        DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
        //获取文件
        File file = new File(path);
}
```

（2）通过 documentBuilderFactory 的 newDocumentBuilder()方法获取 DocumentBuilder 的实例 dombuilder。代码如下：

```
//创建 DocumentBuilder 对象
DocumentBuilder dombuilder = documentBuilderFactory.newDocumentBuilder();
```

（3）使用 dombuilder 的 parse()方法即可读取 XML 文件了。代码如下：

```
dombuilder.parse(file);                            //解析 XML 文件
```

秘笈心法

心法领悟 024：parse()方法的异常处理。

使用 parse()方法时，需要抛出 3 个异常处理，即 ParserConfigurationException、SAXException 和 IOException。异常处理有两种方式：可以使用 throws 直接抛出，留给后面需要调用 parseReadFile()的方法进行处理；也可以直接在本方法中使用 try/catch 把错误包容在本方法中。如下所示：

```
DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory
        .newInstance();                                           //创建 DocumentBuilderFactory 的实例 documentBuilderFactory
DocumentBuilder dombuilder = null;
File file = new File(path);                                       //获取文件
try {
    dombuilder = documentBuilderFactory.newDocumentBuilder();     //获取到 DocumentBuilder 的实例 dombuilder
    dombuilder.parse(file);                                       //解析文件
     } catch (ParserConfigurationException e) {
    //TODO Auto-generated catch block
    e.printStackTrace();
    } catch (SAXException e) {
    //TODO Auto-generated catch block
    e.printStackTrace();
    } catch (IOException e) {
    //TODO Auto-generated catch block
    e.printStackTrace();
    }
```

实例 025 使用 DOM 组件从数据流中读取 XML

光盘位置：光盘\MR\01\025

初级

实用指数：★★★★

DOM 也可以从数据流中读取 XML。当 XML 被保存到数据库中或者使用其他方式存储时，可以使用数据流的方式读取。在本实例中将以 InputStream 的方式读取 XML，在 main()方法中把 XML 存放的路径传给

parseInputStream()方法。要读取的 XML 文件如图 1.25 所示。

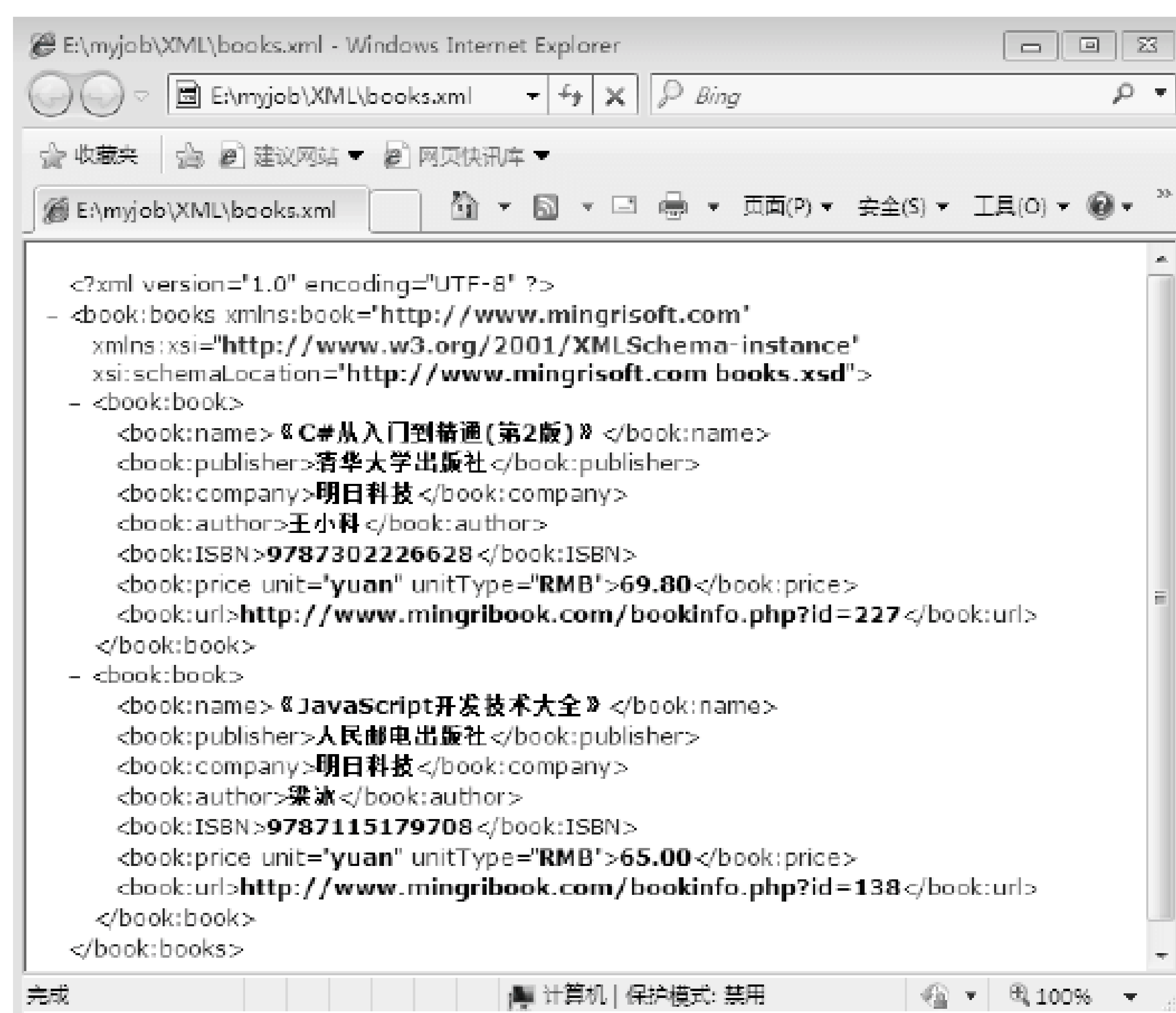


图 1.25　使用 DOM 组件从数据流中读取 XML

使用 DOM 解析 XML 文件前，先通过 DocumentBuilderFactory 的 newInstance()方法创建一个 DocumentBuilderFactory 的实例 documentBuilderFactory，通过 documentBuilderFactory 可以获取 DocumentBuilder 的实例 dombuilder，然后使用 dombuilder 的 parse()方法解析 File 的对象。代码如下：

```
package com.mingrisoft.DOM_demo;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.xml.sax.SAXException;
public class ParserInputStream {
/**
	 * 使用 InputStream 读取 XML 文件
	 *
	 * @param path
	 * @throws ParserConfigurationException
	 * @throws SAXException
	 * @throws IOException
	 */
		public void parseInputStream(String path)
			throws ParserConfigurationException, SAXException, IOException {
		}
		public static void main(String[] arg) {
			ParserInputStream parserFile = new ParserInputStream();
			String path = "D:/eclipseWorkspace/second/xmldemo/books.xml";
			try {
				parserFile.parseInputStream(path);
			} catch (ParserConfigurationException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			} catch (SAXException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}
}
```

设计过程

（1）使用 DocumentBuilderFactory 的 newInstance()方法创建实例 documentBuilderFactor。代码如下：

```
DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory
                .newInstance();                                    //创建 DocumentBuilderFactory 对象
```

（2）使用 documentBuilderFactory 的 newDocumentBuilder()方法获取 DocumentBuilder 的实例 dombuilder。代码如下：

```
DocumentBuilder dombuilder = documentBuilderFactory
                .newDocumentBuilder();                             //创建 DocumentBuilder 对象
```

（3）使用 FileInputStream 类实例化 InputStream，使用参数 path 作为 XML 文件的路径，然后调用 dombuilder 的 parse()方法即可读取 XML 文件。代码如下：

```
InputStream is = new FileInputStream(path);
dombuilder.parse(is);                                              //实例化 InputStream
```

秘笈心法

心法领悟 025：InputStream 接口的说明。

使用 DOM 解析 XML 文件时，可以读取数据流 InputStream。实例中使用 FileInputStream 实例化 InputStream 只是其中一种途径，因为 FileInputSteam 实现了 InputStream 接口，任何实现 InputStream 接口的实体类都可以获取 InputStream。

实例 026 使用 JDOM 组件从文件中读取 XML

光盘位置：光盘\MR\01\026

初级

实用指数：★★★★

实例说明

使用 JDOM 解析 XML，首先要读取 XML 文件。一般对文件的操作都是使用 File 类，把文件的访问路径通过 File 的构造方法传递进去得到一个 File 的对象，然后对这个 File 对象进行读取。本例要读取的 XML 文件如图 1.26 所示。

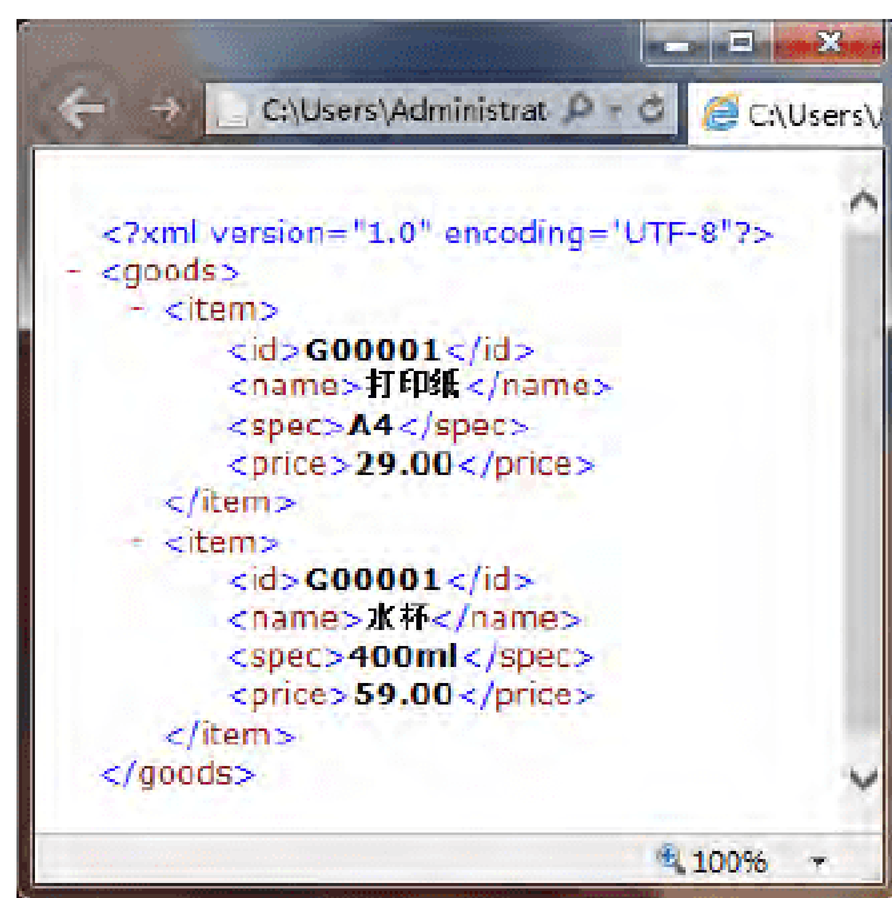


图 1.26　使用 JDOM 组件从文件中读取 XML

关键技术

在使用 JDOM 解析 XML 时，首先需要创建一个 SAXBuilder 解析器，然后通过这个解析器的 build()方法获取一个 Document 对象，这个 Document 对象中就包括了全部 XML，我们可以通过 Document 对象提供的相应方法获取到 XML 的各个节点及属性。在本实例中，主要通过 SAXBuilder 对象的 build()方法来获取 XML，该方法的基本语法格式如下：

```
Document org.jdom.input.SAXBuilder.build(File file) throws JDOMException, IOException
```

参数说明

❶ Document：表示返回值类型为 Document 对象。

❷ file：表示要读取的 File 对象。

❸ throws JDOMException, IOException：表示需要抛出 JDOMException 和 IOException 异常。

（1）创建一个解析器，代码如下：

```
SAXBuilder builder = new SAXBuilder(
"org.apache.xerces.parsers.SAXParser");                  //创建一个解析器
```

（2）根据传递的路径创建一个文件对象，代码如下：

```
File file = new File(path);                              //创建文件对象
```

（3）首先调用 SAXBuilder 解析器的 build()方法通过文件获取对应的 Document 对象，然后获取根节点的元素名并输出。在实现这个步骤时，需要捕获异常，具体代码如下：

```
try {
        org.jdom.Document doc=builder.build(file);                    //通过文件对象获取 Document 对象
        System.out.println("根节点："+doc.getRootElement().getName());  //获取根节点的元素名
} catch (JDOMException e) {
        e.printStackTrace();
}
```

秘笈心法

心法领悟 026：使用 JDOM 解析字符串形式的 XML。

通过 JDOM 组件不仅可以从文件中读取 XML，也可以读取 WebService 接口返回的一段 XML 字符串。代码如下：

```
String xml="<?xml version=\"1.0\" encoding=\"UTF-8\"?><goods><item><id>G00001</id><name>打印纸</name> <spec>A4</spec><price>
                29.00</price></item></goods>";
StringReader xmlReader=new StringReader(xml);                        //创建一个新的字符串
InputSource xmlSource=new InputSource(xmlReader);                    //创建新的输入源 SAX
SAXBuilder builder1 = new SAXBuilder();                              //创建一个解析器
try {
        org.jdom.Document doc=builder1.build(xmlSource);             //通过输入源 SAX 构造一个 Document 对象
        System.out.println("根节点："+doc.getRootElement().getName()); //获取根节点的元素名
} catch (JDOMException e) {
        e.printStackTrace();
}
```

实例 027	使用 JDOM 组件读取 XML 光盘位置：光盘\MR\01\027	高级 实用指数：★★★★☆

实例说明

下面以一个图书网站为例，演示如何读取存储图书信息的 XML 文件，了解 JDOM 简单易用的特性。程序运行结果如图 1.27 所示。



图 1.27　使用 JDOM 组件读取 XML

下面简单介绍一些类的使用方法。

（1）Document 类的使用方法

在 JDOM 中，Document 的使用比 DOM 要简单得多。例如：

```
Element root = new Element("MingRi");
Document doc = new Document(root);
```

```
root.setText("MingRiSoft");
```

上面 3 行代码如果在 DOM 中使用会繁琐很多。例如：

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.newDocument();
Element root = doc.createElement("root");
Text text = doc.createText("ROOT");
root.appendChild(text);
doc.appendChild(root);
```

使用 JDOM 时，采用的 Document 类是 org.JDOM.Document 对象，而不是 org.w3c.dom 中的 Document 类。JDOM 不允许同一个节点同时被两个或多个文档相关联，要在第 2 个文档中使用原来文档的节点，需要使用 detach()方法把这个节点分开。

DOM 的 Document 和 JDOM 的 Document 之间的相互转换方法如下：

```
DOMBuilder builder = new DOMBuilder();
org.jdom.Document jdomDocument = builder.build(domDocument);
DOMOutputter converter = new DOMOutputter();//实例化为一个 DOMOutputter 对象
org.w3c.dom.Document domDocument = converter.output(jdomDocument);
```

（2）Element 类的使用方法

浏览 Element 树可以使用如下方法：

```
Element root = doc.getRootElement();                    //获得根元素
List allChildren = root.getChildren();                  //获得所有子元素
List namedChildren = root.getChildren("name");          //获得指定名称的所有子元素
Element child = root.getChild("name");                  //获得指定名称的第 1 个子元素
List allChildren = root.getChildren();                  //获得元素点的所有子元素
```

删除 Element 元素的方法如下：

```
allChildren.removeAll(root.getChildren("mr"));          //删除所有名为 mr 的子元素
root.removeChildren("mr");                              //删除所有名为 mr 的子元素
allChildren.remove(1);                                  //删除第 2 个子元素
```

添加新元素的方法如下：

```
allChildren.add(new Element("mr"));                     //添加新元素
root.addContent(new Element("mr"));                     //添加新元素
allChildren.add(0, new Element("first"));               //添加新的元素到第 1 个位置
```

读取 Element 的 text 内容可以使用 element.getText()方法或者 element.getTextTrim()方法。另外，可以使用 element.setText()方法修改 Element 的内容。

（3）CDATA 数据操作

```
element.addContent(new CDATA("<xml> content"));
String noDifference = element.getText();
```

■

（1）首先创建 XML 文件 book.xml，在其中存储两本图书信息，其中包括图书的编号、书名、出版社和单价。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
    <book>
        <id>97871151416892</id>
        <name>JSP 数据库系统开发案例精选</name>
        <publish>人民邮电出版社</publish>
        <price>49.00</price>
    </book>
    <book>
        <id>9787115145475</id>
        <name>JSP 数据库开发完全手册</name>
        <publish>人民邮电出版社</publish>
        <price>52.00</price>
    </book>
</books>
```

（2）创建 index.jsp 首页文件，导入需要的 Java 和 JDOM 类库，例如 java.util.*、org.jdom.*等。程序代码如下：

```
<%@ page language="java" pageEncoding="GB18030"%>
<%@ page
    import="java.io.File,
            java.util.*,
            org.jdom.*,
            org.jdom.input.*,
            org.jdom.output.*"%>
<HTML>
    <HEAD>
        <TITLE>长春欣欣电子商城</TITLE>
        <META http-equiv=Content-Type content="text/html; charset=gb2312">
        <LINK href="images/style.css" rel=stylesheet>
        <META content="MSHTML 6.00.3790.1830" name=GENERATOR>
    </HEAD>
    <BODY>
        <CENTER>
            <TABLE cellSpacing=0 cellPadding=0 width=792 align=center border=0>
……
<TABLE cellSpacing=0 cellPadding=0 width="100%"
    border=0>
    <TBODY>
        <TR>
            <TD width="98%" background=images/bg_10.jpg height=35></TD>
        </TR>
        <TR vAlign=top align=middle>
            <TD colSpan=2 height=134>
                <TABLE height=162 cellSpacing=0 cellPadding=0
                width="100%" border=0>
                    <TR>
```

（3）使用 SAXBuilder 解析 XML 文件，获取 XML 文档的根元素，再通过根元素获取文档的节点数据，在循环中遍历 XML 文件中所有图书信息。代码如下：

```
   <%
                    String xmlFile = application.getRealPath("xml/book.xml");
                        SAXBuilder builder = new SAXBuilder(
                            "org.apache.xerces.parsers.SAXParser");
                        Document doc = builder.build(xmlFile);
                        Element root = doc.getRootElement();
                        List nodes = root.getChildren();
                        Iterator iterator = nodes.iterator();
                        while (iterator.hasNext()) {
                            Element node = (Element) iterator.next();
                    %>
                    <TD vAlign=top width="49%" height=162>
                        <TABLE cellSpacing=0 cellPadding=0
……//省略部分代码
                <IMG height=20 src="images/MORE.GIF" width=50
                    border=0>
            </TD>
            <TD width="2%">
                 
            </TD>
        </TR>
    </TBODY>
</TABLE>
```

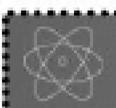
心法领悟 027：JDOM 的优点与不足。

（1）JDOM 是专用于 Java 技术的，比 DOM 应用占用更少的内存。

（2）JDOM 提供了更简单、更具逻辑性的访问 XML 信息的基础方法，但同时也失去了一些灵活性。

（3）JDOM 在处理比较大的 XML 文件时，可能会受到内存限制，只有 XML 文档的大小不超出 RAM 内存容量，才能被 JDOM 解析。

（4）除 XML 文件外，JDOM 还可以访问其他数据源，例如，以创建类从 SQL 查询结果中访问数据。



实例 028 使用 SAX 组件从文件中读取 XML

高级

光盘位置：光盘\MR\01\028

实用指数：★★★★★

实例说明

使用 SAX 解析 XML 文件的第一步，就是要使用 SAX 读取 XML 内容。其方法有很多，如果 XML 被存储成文档，并且文档的内容不是特别大，一般都是通过读取 File 进行操作。本例读取的 XML 文件如图 1.28 所示。

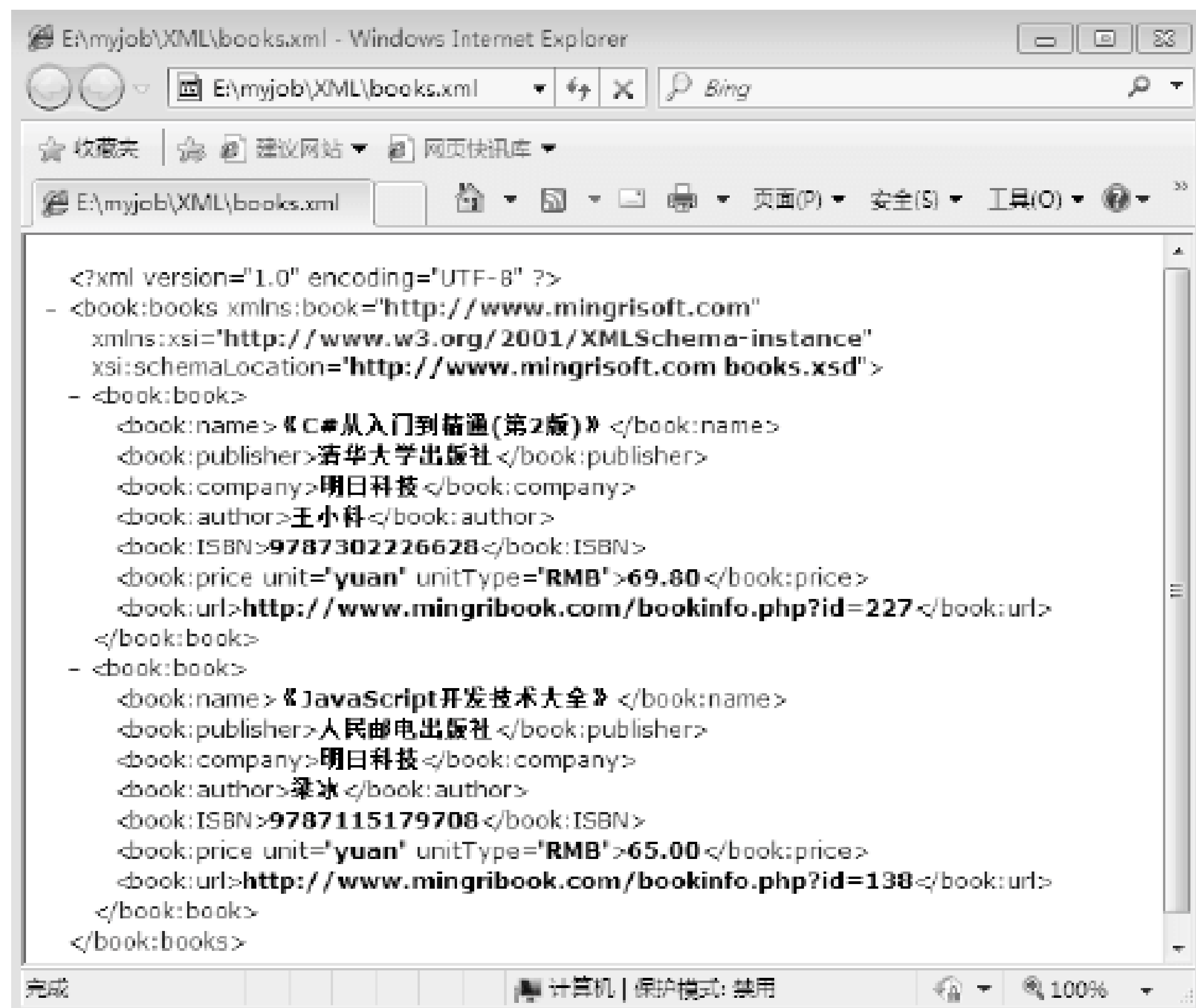


图 1.28　使用 SAX 组件从文件中读取 XML

关键技术

使用 SAX 读取文件时，先通过 SAXParseFactory 的 newInstance()方法创建一个 SAXParseFactory 的实例 factory，然后从 factory 可以获取到 SAXParser 的实例 parser，使用实例 parser 的 parse()方法即可读取 XML 文件。其中 parse()方法有两个参数，一个是解析时需要指出的 XML 的 File 对象，另一个是解析实现的类 DefaultHandler。代码如下：

```
SAXParser parser;
SAXParserFactory factory = SAXParserFactory.newInstance();          //创建 SAXParserFactory 的实例
try {
        parser = factory.newSAXParser();
                File file = new File(pathname);                      //获取文件
                parser.parse(file, new DefaultHandler());            //解析 XML 文件
                } catch (ParserConfigurationException e) {
                //TODO Auto-generated catch block
                e.printStackTrace();
                } catch (SAXException e) {
                //TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IOException e) {
                        //TODO Auto-generated catch block
                        e.printStackTrace();
                }
```

使用 parse()方法时，需要抛出 3 个异常处理 ParserConfigurationException、SAXException 和 IOException，否则程序无法编译通过。

（1）创建一个 XML 文档用于 SAX 读取。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<book:books xmlns:book="http://www.mingrisoft.com"
```

```
<!--声明文档的命名空间-->
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.mingrisoft.com books.xsd">
        <!--声明 XML schema 的存储地址-->
        <book:book>
                <book:name>《C#从入门到精通(第 2 版)》</book:name>
                <book:publisher>清华大学出版社</book:publisher>
                <book:company>明日科技</book:company>
                <book:author>王小科</book:author>
                <book:ISBN>9787302226628</book:ISBN>
                <book:price unit="yuan" unitType="RMB">69.80</book:price>
                <book:url>http://www.mingribook.com/bookinfo.php?id=227</book:url>
        </book:book>
        <book:book>
                <book:name>《JavaScript 开发技术大全》</book:name>
                <book:publisher>人民邮电出版社</book:publisher>
                <book:company>明日科技</book:company>
                <book:author>梁冰</book:author>
                <book:ISBN>9787115179708</book:ISBN>
                <book:price unit="yuan" unitType="RMB">65.00</book:price>
                <book:url>http://www.mingribook.com/bookinfo.php?id=138</book:url>
        </book:book>
</book:books>
```

（2）创建一个 Java 文件，在其中创建 parseReadFile()方法，然后分别创建 SAXParseFactory 的实例 factory 和 File 的实例 file。设置 File 实例时需要用到 XML 文档路径 pathname，pathname 可以通过本方法的参数传递进来。具体代码如下：

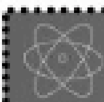
```
public void parseReadFile(String pathname) {
                SAXParser parser;
                SAXParserFactory factory = SAXParserFactory.newInstance();            //创建 SAXParseFactory 的实例 factory
                                try {
                                parser = factory.newSAXParser();
                                File file = new File(pathname);                        //获取文件
                                parser.parse(file, new DefaultHandler());
                        } catch (ParserConfigurationException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } catch (SAXException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
        }
```

（3）创建 main()方法，使用 parser 的 parse()方法即可读取 XML 的文件内容。代码如下：

```
public static void main(String[] arg) {
                        String pathname = "D:/eclipseWorkspace/second/xmldemo/books.xml";
                        new ParserFile().parseReadFile(pathname);
                        ……
        }
```

心法领悟 028：使用 parse()方法时的两个参数。

parser 的 parse()方法有两个参数：一个是 File 的实例 file，file 是 SAX 需要读取的 XML 文件；另一个是 DefaultHandler 的实例，DefaultHandler 是 SAX 中一个没有任何实现的类，其作用是调用 parser 的 parse()方法读取 XML 文件时对文件进行解析（它继承了 EntityResolver、DTDHandler、ContentHandler、ErrorHandler 4 个接口）。虽然在此使用了 parser 的 parse()方法，但是 DefaultHandler 是一个没有实现的类，所以如果希望使用 SAX 解析，还要重新实现 DefaultHandler。



实例 029 使用 SAX 组件从数据流中读取 XML

光盘位置：光盘\MR\01\029

高级

实用指数：★★★★★

实例说明

如果 XML 被存储成文件的格式，SAX 将按文件的方式读取 XML；如果 XML 被保存到数据库中或者使用其他方式存储，则可以使用数据流的方式读取。在本实例中 SAX 将借助 InputStream 读取 XML。为了更方便地操作 InputStream，可以在 main()方法中使用 File 得到 InputStream。读取的 XML 文件如图 1.29 所示。

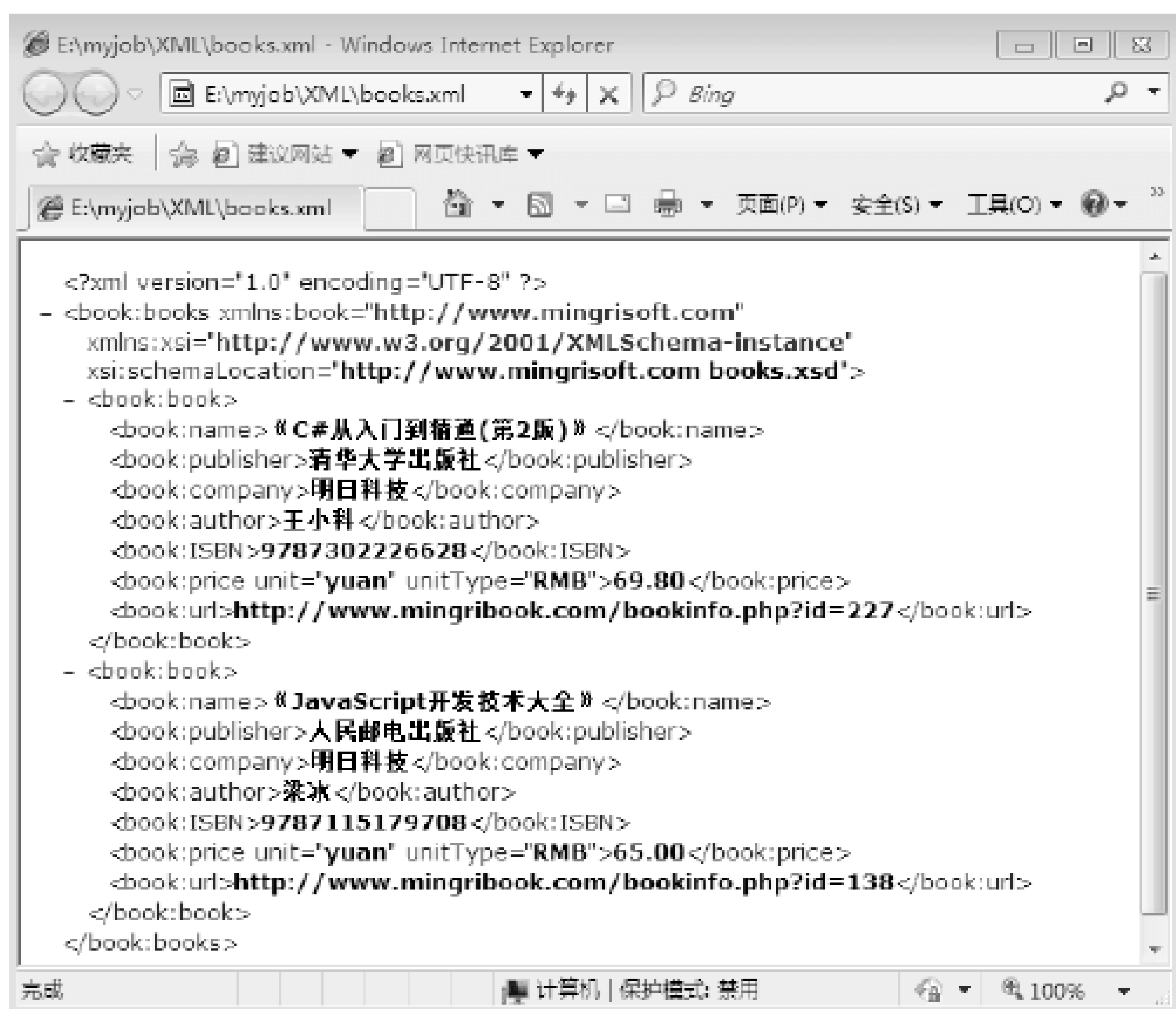


图 1.29　使用 SAX 组件从数据流中读取 XML

关键技术

使用 SAXParserFactory 的 parse()方法可以从数据流中读取 XML。语法如下：

```
public void parse(InputStream is, DefaultHandler dh) throws SAXException, IOException
```

参数说明

❶ is：表示要读取的 XML 数据流。

❷ dh：表示读取的 XML 文件处理机制。

（1）创建一个 XML 文档用于 SAX 读取。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<book:books xmlns:book="http://www.mingrisoft.com"
<!--声明文档的命名空间-->
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.mingrisoft.com books.xsd">
        <!--声明 XML schema 的存储地址-->
        <book:book>
                <book:name>《C#从入门到精通(第 2 版)》</book:name>
                <book:publisher>清华大学出版社</book:publisher>
                <book:company>明日科技</book:company>
                <book:author>王小科</book:author>
                <book:ISBN>9787302226628</book:ISBN>
                <book:price unit="yuan" unitType="RMB">69.80</book:price>
                <book:url>http://www.mingribook.com/bookinfo.php?id=227</book:url>
        </book:book>
        <book:book>
                <book:name>《JavaScript 开发技术大全》</book:name>
```

```
            <book:publisher>人民邮电出版社</book:publisher>
            <book:company>明日科技</book:company>
            <book:author>梁冰</book:author>
            <book:ISBN>9787115179708</book:ISBN>
            <book:price unit="yuan" unitType="RMB">65.00</book:price>
            <book:url>http://www.mingribook.com/bookinfo.php?id=138</book:url>
        </book:book>
</book:books>
```

（2）创建一个 Java 文件，在其中创建 parseInputStream()方法，然后通过 SAXParseFactory 类的 newSAXParser()方法获取 SAXParser 的实例，再获取 InputStream 和 DefaultHandler 的对象，最后使用 SAXParser 的 parse()方法对数据流进行读取、解析。代码如下：

```
public void parseInputStream(InputStream inputStream) {
        SAXParser parser;                              //生成 SAXParserFactory 实例
        SAXParserFactory factory = SAXParserFactory.newInstance();
        try {
                parser = factory.newSAXParser();       //获取 SAXParser 实例
                parser.parse(inputStream, new DefaultHandler());
        } catch (ParserConfigurationException e) {
                e.printStackTrace();
        } catch (SAXException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

（3）创建 main()方法，在该方法中先通过 File 和 FileInputStream 创建 InputStream 的对象，然后调用 parseInputStream()方法实现 SAX 通过数据流读取 XML 内容。代码如下：

```
public static void main(String[] arg) {
        String pathname = "xmldemo/books.xml";
        InputStream inputStream = null;
        try {
                //获取数据流
                inputStream = new FileInputStream(new File(pathname));
        } catch (FileNotFoundException e1) {
                e1.printStackTrace();
        }
        //读取文件流
        new ParserInputStream().parseInputStream(inputStream);
        try {
                //关闭文件流
                inputStream.close();
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}
```

秘笈心法

心法领悟 029：数据流的关闭。

使用 XML 读取 InputStream 时是以流的方式进行的，在使用完 InputStream 以后要调用 close()方法关闭数据流，避免程序占用系统资源。

实例 030	使用 DOM 组件解析 XML 元素名称 光盘位置：光盘\MR\01\030	高级 实用指数：★★★

实例说明

解析 XML 时，首先要获取元素名称。DOM 把元素信息保存在 Node 里，通过 Node 类的 getNodeName()方

法可以获取元素名称。本实例将演示如何使用 DOM 解析 XML 元素名称，如图 1.30 所示。

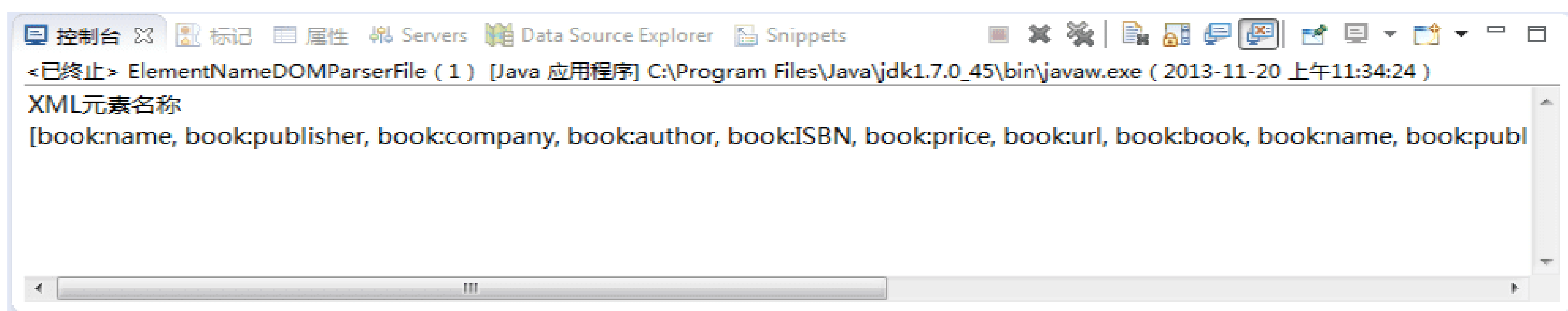


图 1.30 使用 DOM 组件解析 XML 元素名称

关键技术

（1）使用 Node 类的 getChildNodes()方法可以获取 XML 文件中的子元素列表。语法如下：

```
public NodeList getChildNodes()
```

（2）使用 Node 类的 hasChildNodes()方法可以判断当前 XML 元素是否还有子节点。语法如下：

```
public boolean hasChildNodes()
```

（1）创建一个 XML 文档用于 DOM 解析。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<book:books xmlns:book="http://www.mingrisoft.com"
<!--声明文档的命名空间-->
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.mingrisoft.com books.xsd">
        <!--声明 XML schema 的存储地址-->
        <book:book>
                <book:name>《C#从入门到精通(第 2 版)》</book:name>
                <book:publisher>清华大学出版社</book:publisher>
                <book:company>明日科技</book:company>
                <book:author>王小科</book:author>
                <book:ISBN>9787302226628</book:ISBN>
                <book:price unit="yuan" unitType="RMB">69.80</book:price>
                <book:url>http://www.mingribook.com/bookinfo.php?id=227</book:url>
        </book:book>
        <book:book>
                <book:name>《JavaScript 开发技术大全》</book:name>
                <book:publisher>人民邮电出版社</book:publisher>
                <book:company>明日科技</book:company>
                <book:author>梁冰</book:author>
                <book:ISBN>9787115179708</book:ISBN>
                <book:price unit="yuan" unitType="RMB">65.00</book:price>
                <book:url>http://www.mingribook.com/bookinfo.php?id=138</book:url>
        </book:book>
</book:books>
```

（2）创建 ElementNameDOMParserFile 类的 parseReadFile()方法，用于读取 XML 文件，同时返回一个 Document。代码如下：

```
public Document parseReadFile(String path) throws ParserConfigurationException, SAXException, IOException {
        DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory
                        .newInstance();                         //创建 DocumentBuilderFactory 实例
        DocumentBuilder dombuilder = documentBuilderFactory
                        .newDocumentBuilder();                  //创建 DocumentBuilder 实例
        File file = new File(path);                             //获取 XML 文件
        return dombuilder.parse(file);                          //解析 XML 文件
}
```

（3）创建 getElementName()方法，通过其参数 Node 的 hasChildNodes()方法和 getChildNodes()方法遍历 Node 的内容。当 Node 存在子节点时，使用 getNodeName()方法得到当前节点的名称（也就是元素的名称），同时保存在 List 中，然后递归调用 getElementName()方法获取下级子节点的元素名称（因为 XML 文

档就是一个树形结构，只有这样才能得到所有 XML 元素的名称）。代码如下：

```
public List<String> getElementName(Node parentNode) {
        //是否有子节点
        if (parentNode.hasChildNodes()) {
            //获取子节点
            NodeList nodeList = parentNode.getChildNodes();
            for (int i = 0; i < nodeList.getLength(); i++) {
                Node node = nodeList.item(i);
                //如果有子节点则递归调用
                if (node.hasChildNodes()) {
                    getElementName(node);
                    elementList.add((node.getNodeName()));
                }
            }
        }
        return elementList;
    }
```

（4）创建 main()方法，在其内部调用 parseReadFile()和 getElementName()方法，最后把结果输出到控制台。代码如下：

```
public static void main(String[] arg) {
        ElementNameDOMParserFile parserFile = new ElementNameDOMParserFile();
        String path = "xmldemo/books.xml";
        try {
            Document document = parserFile.parseReadFile(path);              //创建 DocumentBuilder 实例
            List<String> list = parserFile.getElementName(document);        //将结果保存到 list 中
            System.out.println("XML 元素名称");
            System.out.println(list);                                       //将结果输出到控制台
        } catch (ParserConfigurationException e) {
            //TODO Auto-generated catch block
            e.printStackTrace();
        } catch (SAXException e) {
            //TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            //TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 030：XML 元素名称的获取。

在 main()方法中根据 parseReadFile 得到一个 Document，因为 Document 继承了 Node 接口，而 getElementName()的参数是 Node，所以可以把 Document 对象传入到 getElementName()方法中，由 getElementName()方法负责获取 XML 元素名称。

实例 031	使用 DOM 组件解析 XML 元素名称和内容 光盘位置：光盘\MR\01\031	高级 实用指数：★★★

实例说明

XML 元素的名称和内容都可以使用 DOM 解析出来。本实例就实现了使用 DOM 解析 XML 元素名称和内容，然后把它们输出到控制台，如图 1.31 所示。

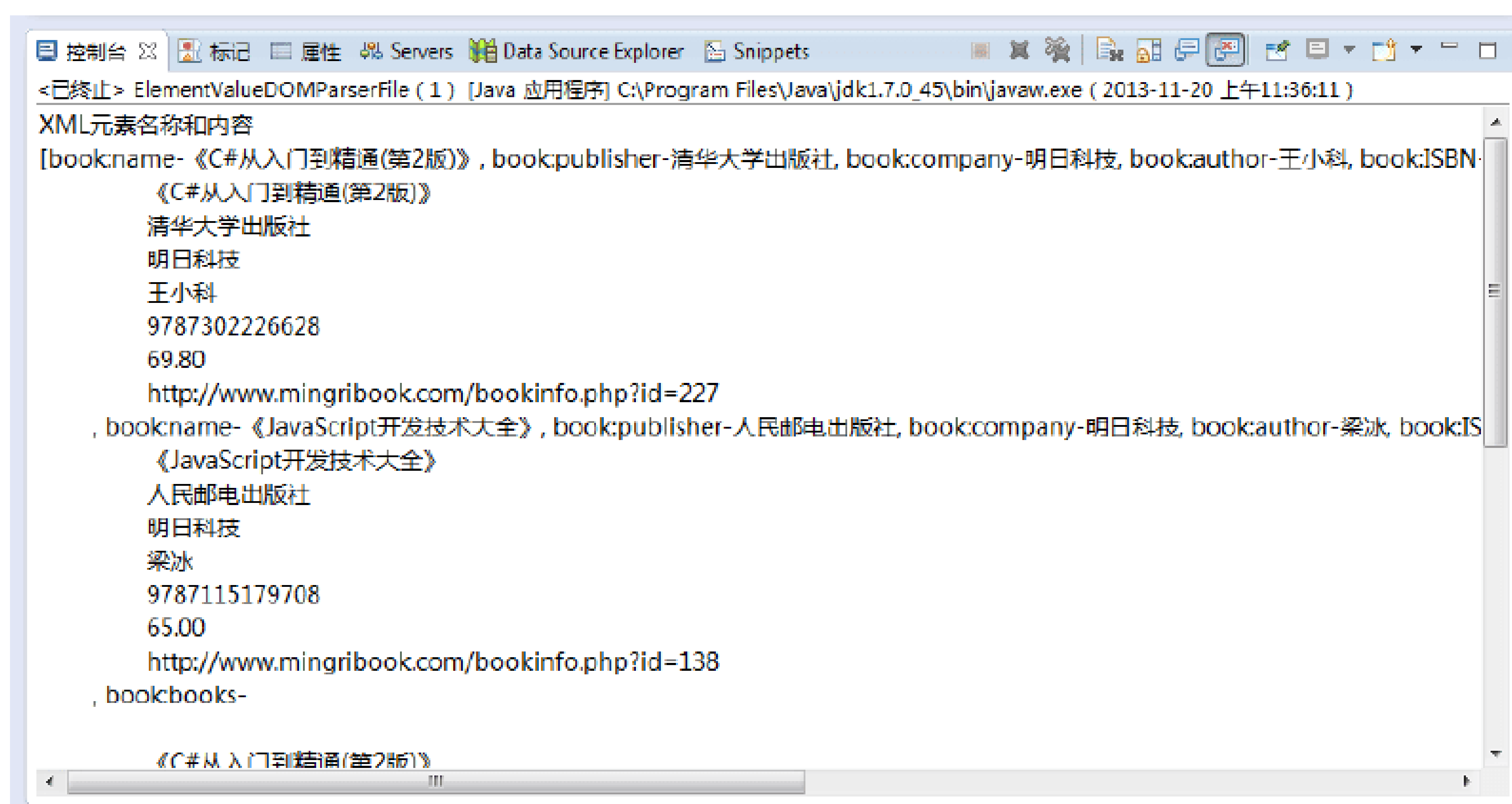


图 1.31　使用 DOM 组件解析 XML 元素名称和内容

关键技术

使用 Node 类的 getNodeName()方法可以获取元素节点的名称。语法如下：

```
public String getNodeName()
```

使用 Node 类的 getTextContent()方法可以获取元素内容。语法如下：

```
public String getTextContent() throws DOMException;
```

（1）创建一个 XML 文档用于 DOM 解析。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<book:books xmlns:book="http://www.mingrisoft.com"
<!--声明文档的命名空间-->
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.mingrisoft.com books.xsd">
	<!--声明 XML schema 的存储地址-->
	<book:book>
		<book:name>《C#从入门到精通(第 2 版)》</book:name>
		<book:publisher>清华大学出版社</book:publisher>
		<book:company>明日科技</book:company>
		<book:author>王小科</book:author>
		<book:ISBN>9787302226628</book:ISBN>
		<book:price unit="yuan" unitType="RMB">69.80</book:price>
		<book:url>http://www.mingribook.com/bookinfo.php?id=227</book:url>
	</book:book>
	<book:book>
		<book:name>《JavaScript 开发技术大全》</book:name>
		<book:publisher>人民邮电出版社</book:publisher>
		<book:company>明日科技</book:company>
		<book:author>梁冰</book:author>
		<book:ISBN>9787115179708</book:ISBN>
		<book:price unit="yuan" unitType="RMB">65.00</book:price>
		<book:url>http://www.mingribook.com/bookinfo.php?id=138</book:url>
	</book:book>
</book:books>
```

（2）创建 ElementValueDOMParserFile 类，在该类中创建 parseReadFile()方法用于读取 XML 文件，同时返回一个 Document。代码如下：

```
public class ElementValueDOMParserFile {
	public Document parseReadFile(String path)                    //读取 XML 文件
				hrows ParserConfigurationException, SAXException, IOException {
			DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory
				.newInstance();                                   //创建 DocumentBuilderFactory 实例
			DocumentBuilder dombuilder = documentBuilderFactory
```

```
                    .newDocumentBuilder();                          //获取 DocumentBuilder 实例
            File file = new File(path);
            return dombuilder.parse(file);
    }
}
```

（3）建立一个 List 变量，用于保存 XML 文件的元素和内容。代码如下：

```
private List<String> elementList = new ArrayList<String>();
```

（4）创建一个 getElementName()方法，在该方法内部读取 XML 文件的元素和内容，然后拼成一个字符串，再保存到 List 的变量里。代码如下：

```
public void getElementName(Node parentNode) {
    if (parentNode.hasChildNodes()) {
        NodeList nodeList = parentNode.getChildNodes();
        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            //判断是否有子节点
            if (node.hasChildNodes()) {
                getElementName(node);
                //拼成字符串，保存在 List 中
                elementList.add(node.getNodeName() + "-"+ node.getTextContent());
            }
        }
    }
}
```

（5）创建 getElementList()方法，其返回保存在 List 变量中的结果。代码如下：

```
public List<String> getElementList() {
    return this.elementList;                                        //返回 List 中的结果
}
```

（6）创建 main()方法，在其内部调用刚才创建的方法，最后把结果输出到控制台。代码如下：

```
public static void main(String[] arg) {
        ElementValueDOMParserFile parserFile = new ElementValueDOMParserFile();
        String path = "xmldemo/books.xml";
        try {
            Document document = parserFile.parseReadFile(path);     //创建 Document 的实例
            parserFile.getElementName(document);
            List<String> list = parserFile.getElementList();
            System.out.println("XML 元素名称和内容");
            System.out.println(list);                               //将结果输出到控制台
        } catch (ParserConfigurationException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (SAXException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
```

■

心法领悟 031：父节点的内容。

如果通过 DOM 获取一个父节点的内容，那么这个父节点的内容就是其所有子节点内容的集合。例如，实例中第一个 book:book 元素下的子节点有 book:name、book:publisher、book:company、book:author、book:ISBN、book:price 和 book:url，这些元素的内容分别是《C#从入门到精通(第 2 版)》、清华大学出版社、明日科技、王小科、9787302226628、69.80、http://www.mingribook.com/bookinfo.php?id=227，那么 book:book 元素的内容就是它们的集合。

```
《C#从入门到精通(第 2 版)》
        清华大学出版社
        明日科技
        王小科
        9787302226628
        69.80
        http://www.mingribook.com/bookinfo.php?id=227
```

实例 032 使用 SAX 组件解析 XML 元素名称

光盘位置：光盘\MR\01\032

高级

实用指数：★★★

实例说明

本实例实现了使用 SAX 组件解析 XML 元素名称，并把解析的 XML 元素名称打印到控制台，如图 1.32 所示。

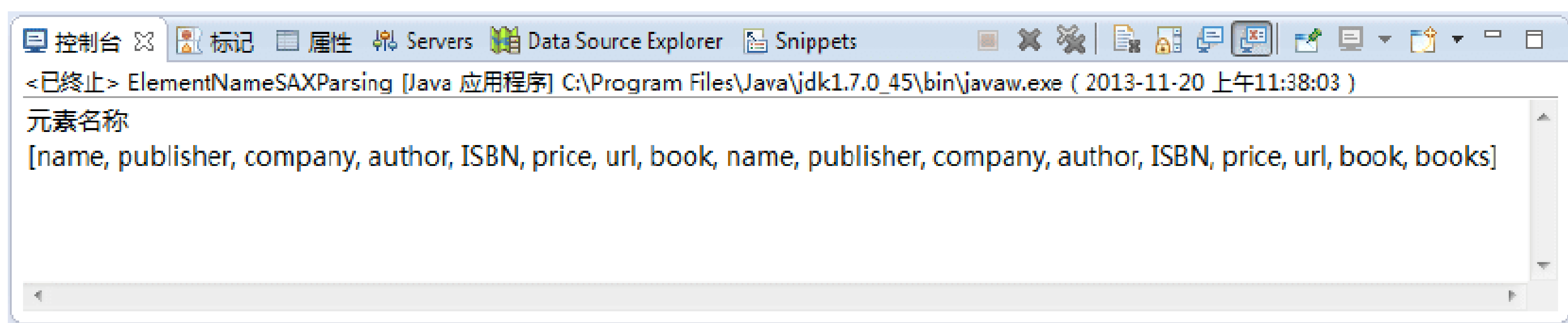


图 1.32 使用 SAX 组件解析 XML 元素名称

关键技术

SAX 解析 XML 时按顺序以流的方式读取 XML。在此用 ElementNameSAXParsing 继承了 DefaultHandler 类，并重写 endElement()方法。以后每次调用 SAXParser 的 parse()方法，向 parse()方法中传入 XML 文件和 ElementNameSAXParsing 的实例，SAX 读取 XML 元素结束时 endElement()方法就会被执行。在 XML 文件中有几个 XML 元素，解析结束时 endElement()就会被调用几次。endElement()方法的语法如下：

```
public void endElement(String uri, String localName, String qName)  throws SAXException
```

参数说明

❶ uri：表示 XML 元素命名空间，在此处为 http://www.mingrisoft.com。

❷ localName：表示 XML 元素的本地标识符，此处为 name、publisher、company、author、ISBN 等。

❸ qName：表示元素在 XML 文件中使用的名称，此处为 book:name、book:publisher、book:company、book:author、book:ISBN 等。

设计过程

（1）创建一个 XML 文档用于 SAX 解析。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<book:books xmlns:book="http://www.mingrisoft.com"
<!--声明文档的命名空间-->
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.mingrisoft.com books.xsd">
        <!--声明 XMLSchema 的存储地址-->
        <book:book>
                <book:name>《C#从入门到精通(第 2 版)》</book:name>
                <book:publisher>清华大学出版社</book:publisher>
                <book:company>明日科技</book:company>
                <book:author>王小科</book:author>
                <book:ISBN>9787302226628</book:ISBN>
                <book:price unit="yuan" unitType="RMB">69.80</book:price>
                <book:url>http://www.mingribook.com/bookinfo.php?id=227</book:url>
        </book:book>
        <book:book>
                <book:name>《JavaScript 开发技术大全》</book:name>
                <book:publisher>人民邮电出版社</book:publisher>
                <book:company>明日科技</book:company>
                <book:author>梁冰</book:author>
                <book:ISBN>9787115179708</book:ISBN>
                <book:price unit="yuan" unitType="RMB">65.00</book:price>
                <book:url>http://www.mingribook.com/bookinfo.php?id=138</book:url>
```

```
        </book:book>
</book:books>
```

（2）创建 ElementNameSAXParsing 类，然后继承 DefaultHandler 类，在 Java 文件中创建 parseReadFile()方法用于解析 XML。代码如下：

```
public void parseReadFile(String pathname) {
        SAXParser parser;
        SAXParserFactory factory = SAXParserFactory.newInstance();                //创建 SAXParserFactory 的实例
        try {
                factory.setValidating(true);                                      //验证 XML 的格式是否正确
                factory.setNamespaceAware(true);                                  //是否引入 XML 命名空间

                parser = factory.newSAXParser();
                File file = new File(pathname);                                   //获取文件
                parser.parse(file, this);                                         //解析 XML 文件
        } catch (ParserConfigurationException e) {
                e.printStackTrace();
        } catch (SAXException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

（3）在 ElementNameSAXParsing 类中重写 endElement()方法，在该方法中获取 localName，将其保存在 List 中。代码如下：

```
public void endElement(String uri, String localName, String qName) throws SAXException {
        list.add(localName);                                                      //保存到 List 中
}
```

（4）创建 main()方法，在该方法中使用 ElementNameSAXParsing 解析 books.xml 文件，输出元素名称到控制台。代码如下：

```
public static void main(String[] arg) {
        String pathname = "xmldemo/books.xml";
        ElementNameSAXParsing elementSAXParsing = new ElementNameSAXParsing();
        elementSAXParsing.parseReadFile(pathname);
        System.out.println("元素名称");
        System.out.println(elementSAXParsing.getList());
}
```

秘笈心法

心法领悟 032：parseReadFile()方法的使用。

细心的读者可能注意到 parseReadFile()方法里有如下两行代码：

```
//验证 XML 的格式是否正确
factory.setValidating(true);
//是否引入 XML 命名空间
factory.setNamespaceAware(true);
```

设置 setValidating 为 true 时，表示 SAX 在解析 XML 时会验证 XML 的格式是否正确，如果验证没有通过，后台会报相应的错误。

设置 setNamespaceAware 为 true 时，表示 SAX 在解析 XML 时会引入 XML 命名空间。只有引入命名空间，在解析 XML 时，处理 endElement()等方法中 SAX 才会向 uri、localName 参数中传值。

实例 033	使用 SAX 组件解析 XML 元素名称和内容 光盘位置：光盘\MR\01\033	高级 实用指数：★★★

解析 XML 元素名称和内容，关键是当 SAX 解析 XML 时，把元素的名称和内容及时保存起来；同时，XML 中可能会有很多同名的元素，还要保证元素名称和元素内容能够正确对应。本实例在解析 XML 时，在

endElement()方法中把元素名称和内容拼成一个字符串，然后将其保存在 List 中。有了元素名称和内容的 List，要实现其他的业务逻辑就不难了。在这里只是简单的输出，效果如图 1.33 所示。



图 1.33　使用 SAX 组件解析 XML 元素名称和内容

使用 ElementValueSAXParsing 类继承 DefaultHandler 类可以实现 characters()方法，通过该方法可以获取 XML 元素名称和内容。语法如下：

```
public void characters(char[] ch, int start, int length) throws SAXException
```

参数说明

❶ ch：表示 XML 的内容，其内容是整个 XML 文档。

❷ start：表示当前元素在整个 XML 文档中开始的字节数。

❸ length：表示当前元素本身的字节长度。

利用 SAX 解析 XML 时，可创建 ElementValueSAXParsing 类，继承 DefaultHandler 类，并重写 endElement()方法。以后每次调用 SAXParser 的 parse()方法，向 parse()里传入 XML 文件和 ElementValueSAXParsing 的实例，SAX 读取每个元素内容时 characters()方法就会被执行，读取 XML 元素结束时 endElement()方法就会被调用。代码如下：

```
package com.mingrisoft.SAX_demo;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
public class ElementValueSAXParsing extends DefaultHandler {
		private List<String> list = new ArrayList<String>();
		private String value;
		/**
		* 读取当前元素的内容，过滤制表符、空格符、回车符、换行符
		 */
		@Override
		public void characters(char[] ch, int start, int length)
				throws SAXException {
			// TODO Auto-generated method stub
			value = String.valueOf(ch, start, length);
			value = value.replace("\t", "");
			value = value.replace(" ", "");
			value = value.replace("\n", "");
			value = value.replace("\r", "");
		}
		/**
		 * 读取元素结束，把元素名称和元素内容保存在 Map 中
		 */
		@Override
		public void endElement(String uri, String localName, String qName)
			throws SAXException {
			// TODO Auto-generated method stub
```

```
                list.add(localName + ":" + value);
        }
        public List<String> getList() {
                return this.list;
        }
        /**
         * 通过文件读取 XML
         *
        * @param pathname
         *              文件路径
         */
        public void parseReadFile(String pathname) {
                SAXParser parser;
                SAXParserFactory factory = SAXParserFactory.newInstance();
                try {
                        factory.setValidating(true);
                        factory.setNamespaceAware(true);
                        parser = factory.newSAXParser();
                        File file = new File(pathname);
                        parser.parse(file, this);
                } catch (ParserConfigurationException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (SAXException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }
        public static void main(String[] arg) {
                String pathname = "xmldemo/books.xml";
                ElementValueSAXParsing elementSAXParsing = new ElementValueSAXParsing();
                elementSAXParsing.parseReadFile(pathname);
                System.out.println("元素名称和元素内容");
                System.out.println(elementSAXParsing.getList());
        }
}
```

（1）创建一个 XML 文档用于 SAX 解析。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<book:books xmlns:book="http://www.mingrisoft.com"
<!--声明命名空间 book-->
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.mingrisoft.com books.xsd">
        <!--声明 XML schema 文档的存储地址-->
        <book:book>
                <book:name>《C#从入门到精通(第 2 版)》</book:name>
                <book:publisher>清华大学出版社</book:publisher>
                <book:company>明日科技</book:company>
                <book:author>王小科</book:author>
                <book:ISBN>9787302226628</book:ISBN>
                <book:price unit="yuan" unitType="RMB">69.80</book:price>
                <book:url>http://www.mingribook.com/bookinfo.php?id=227</book:url>
        </book:book>
        <book:book>
                <book:name>《JavaScript 开发技术大全》</book:name>
                <book:publisher>人民邮电出版社</book:publisher>
                <book:company>明日科技</book:company>
                <book:author>梁冰</book:author>
                <book:ISBN>9787115179708</book:ISBN>
                <book:price unit="yuan" unitType="RMB">65.00</book:price>
```

```
            <book:url>http://www.mingribook.com/bookinfo.php?id=138</book:url>
        </book:book>
</book:books>
```

（2）创建 ElementValueSAXParsing 类，然后继承 DefaultHandler 接口，在 Java 文件中创建 parseReadFile() 方法用于解析 XML。代码如下：

```
public class ElementValueSAXParsing extends DefaultHandler {            //创建解析 XML 文件的方法
            private String value;
        /**
         * 通过文件读取 XML
         *
         * @param pathname
         *              文件路径
         */
        public void parseReadFile(String pathname) {
            SAXParser parser;
            SAXParserFactory factory = SAXParserFactory.newInstance();
            try {
                factory.setValidating(true);
                factory.setNamespaceAware(true);
                parser = factory.newSAXParser();
                File file = new File(pathname);
                parser.parse(file, this);
            } catch (ParserConfigurationException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (SAXException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
}
```

（3）实现 characters()方法，获取当前元素的内容，将其保存在临时变量 value 中。代码如下：

```
public void characters(char[] ch, int start, int length) throws SAXException {
        //读取当前元素的内容，过滤制表符、空格符、回车符、换行符
        value = String.valueOf(ch, start, length);
        value = value.replace("\t", "");
        value = value.replace(" ", "");
        value = value.replace("\n", "");
        value = value.replace("\r", "");
}
```

（4）实现 endElement()方法，在该方法中获取 localName 和临时变量 value 值，并把元素的名称和内容一起保存在 List 中。代码如下：

```
public void endElement(String uri, String localName, String qName)
            throws SAXException {
        //读取元素结束，把元素名称和内容保存在 List 中
        list.add(localName + ":" + value);
}
```

（5）创建 main()方法，在该方法里使用 ElementValueSAXParsing 解析 books.xml 文件，输出元素名称到控制台。代码如下：

```
public static void main(String[] arg) {
        String pathname = "xmldemo/books.xml";                              //获取 XML 文件地址
        ElementValueSAXParsing elementSAXParsing = new ElementValueSAXParsing();   //创建 ElementValueSAXParsing 实例
        elementSAXParsing.parseReadFile(pathname);                          //解析 XML 文件
        System.out.println("元素名称和元素内容");
        System.out.println(elementSAXParsing.getList());
}
```

心法领悟 033：屏蔽特殊符号。

在 XML 文档中，book 是一个父元素，内部包含了几个子元素，但是它本身并没有元素内容。在读取时，SAX 会把内部一些不可见的字符读出来，如空格符、回车符等，可以通过字符串的替换把这些字符替换掉。

此外，还可以使用 ignorableWhitespace()方法。此方法获取的元素内容都是没有空格符的，但是要想触发 ignorableWhitespace()方法，则必须让 XML 文档引用 DTD，使用 XML Schema 限定 XML 时 ignorableWhitespace()方法是不会被触发的。



实例 034　使用 SAX 组件解析 XML 元素属性和属性值

光盘位置：光盘\MR\01\034

高级　实用指数：★★★

实例说明

在 XML 中每个元素都可能含有属性。属性是针对元素而言的，包含属性名称和属性值。本实例的 XML 文档中含有两本图书，每本图书都有自己的价格，也就是每个 book 元素中都包含一个 price 子元素，但是其内容可能是不一样的。price 元素包含两个属性，即 unit 和 unitType。每个 price 元素既可以有同样的属性，也可以有不同的属性。本实例实现了获取 XML 元素的属性和属性值，如图 1.34 所示。

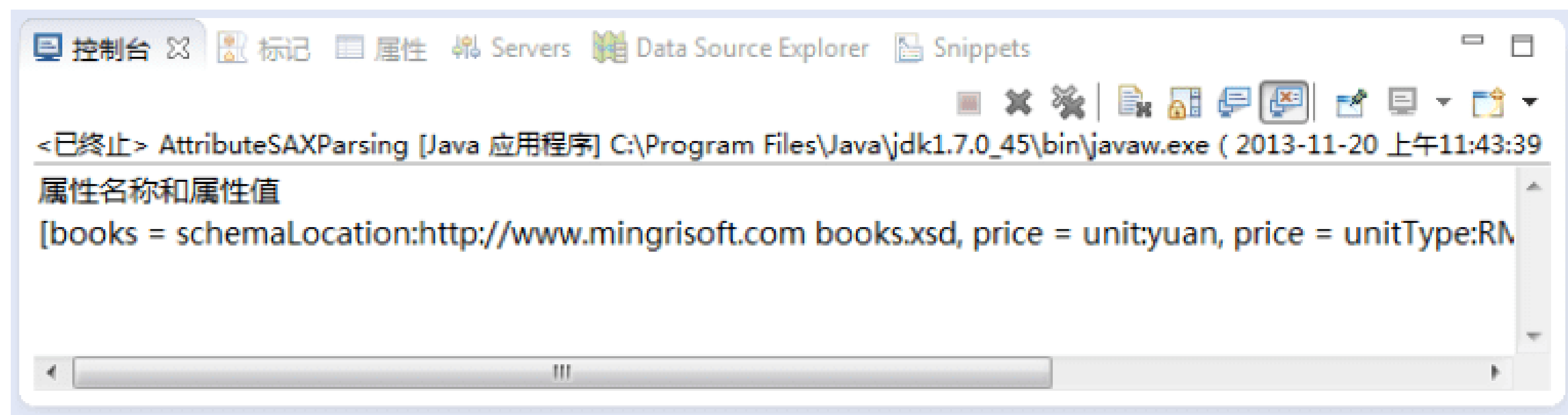


图 1.34　使用 SAX 组件解析 XML 元素属性和属性值

关键技术

SAX 每次开始读取 XML 元素时，startElement()方法都会被执行。使用 AttributeSAXParsing 类重写 DefaultHandler 类的 startElement()方法可以获取元素属性和属性值。语法如下：

```
public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException
```

参数说明

❶ uri：表示 XML 元素命名空间，此处为 http://www.mingrisoft.com。

❷ localName：表示 XML 元素的本地标识符，此处为 name、publisher、company、author、ISBN 等。

❸ qName：表示元素在 XML 文件中使用的名称，此处为 book:name、book:publisher、book:company、book:author、book:ISBN 等。

❹ attributes：表示当前元素的属性集合。

（1）创建一个 XML 文档用于 SAX 解析。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<book:books xmlns:book="http://www.mingrisoft.com"
<!--声明命名空间-->
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.mingrisoft.com books.xsd">
        <!--声明命名空间的存储地址-->
        <book:book>
                <book:name>《C#从入门到精通(第 2 版)》</book:name>
                <book:publisher>清华大学出版社</book:publisher>
                <book:company>明日科技</book:company>
                <book:author>王小科</book:author>
```

```
            <book:ISBN>9787302226628</book:ISBN>
            <book:price unit="yuan" unitType="RMB">69.80</book:price>
            <book:url>http://www.mingribook.com/bookinfo.php?id=227</book:url>
        </book:book>
        <book:book>
            <book:name>《JavaScript 开发技术大全》</book:name>
            <book:publisher>人民邮电出版社</book:publisher>
            <book:company>明日科技</book:company>
            <book:author>梁冰</book:author>
            <book:ISBN>9787115179708</book:ISBN>
            <book:price unit="yuan" unitType="RMB">65.00</book:price>
            <book:url>http://www.mingribook.com/bookinfo.php?id=138</book:url>
        </book:book>
</book:books>
```

（2）创建 AttributeSAXParsing 类，继承 DefaultHandler 类，重写 startElement()方法读取属性和属性值，再把它们拼成一个字符串存储在 List 里。一个元素可能有多个属性，所以使用 attributes 中的 getLength()方法，可以获取当前元素属性的个数。在这个属性集合中，getLocalName()方法可获取当前元素指定属性的名称；getValue()方法中参数是几就表示当前元素的第几个属性值。代码如下：

```
public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {
        //读取属性名称和属性值，保存在 List 中
        for (int i = 0; i < attributes.getLength(); i++) {
            attribute.add(localName + " = " + attributes.getLocalName(i) + ":"+ attributes.getValue(i));
        }
}
```

（3）创建 parseReadFile()方法，把 AttributeSAXParsing 实例传入解析器，实现 XML 的解析。代码如下：

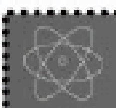
```
public void parseReadFile(String pathname) {
        SAXParser parser;
        SAXParserFactory factory = SAXParserFactory.newInstance();          //获取 SAXParserFactory 实例
        try {
            factory.setValidating(true);
            factory.setNamespaceAware(true);
            parser = factory.newSAXParser();                              //获取 SAXParser 实例
            File file = new File(pathname);                               //获取 XML 文件
            parser.parse(file, this);                                     //解析 XML 文件
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
}
```

（4）创建 main()方法，使用 AttributeSAXParsing 解析 books.xml 文件，输出属性名称和属性值到控制台。代码如下：

```
public static void main(String[] arg) {
        String pathname = "xmldemo/books.xml";
        AttributeSAXParsing elementSAXParsing = new AttributeSAXParsing();
        elementSAXParsing.parseReadFile(pathname);                        //解析 XML 文件
        System.out.println("属性名称和属性值");
        System.out.println(elementSAXParsing.getAttribute());             //输出结果到控制台
}
```

心法领悟 034：使用 startElement()方法获取属性的名称和值。

本实例中使用 startElement()获取了属性的名称和值。attributes 是 startElement()方法的一个参数，通过它可以获取当前元素的所有属性集合和属性其他相关信息。例如，使用 getIndex()方法获取某个属性的索引值，使用 getType()方法获取某个属性的类型，使用 getQName()方法获取某个属性的 XML 元素名称等。



实例 035　使用 DOM 组件解析 XML 元素属性和属性值

光盘位置：光盘\MR\01\035

高级

实用指数：★★★

实例说明

使用 DOM 组件解析 XML 后，每个元素都会转换成一个 Node 对象。Node 内不但存储着元素内容，还保存着元素的属性。使用 getAttributes()方法可以获取当前元素的所有属性。在 XML 中不同父节点的同名子节点的属性有可能是一样的，如本实例中的两个 book:book 元素，它们都有 book:price 子节点，两个子节点的属性都是 unit="yuan"和 unitType="RMB"，使用 DOM 解析 XML 属性如图 1.35 所示。

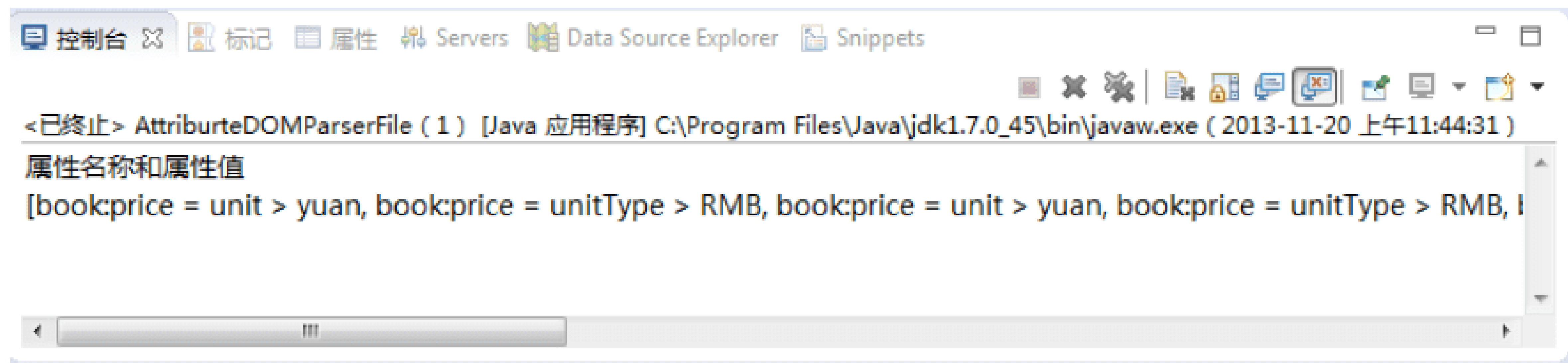


图 1.35　使用 DOM 组件解析 XML 元素属性和属性值

关键技术

（1）使用 Node 类的 getAttributes()方法可以获取属性列表。语法如下：

```
public NamedNodeMap getAttributes()
```

（2）使用 NamedNodeMap 类的 item()方法可以获取指定的节点。语法如下：

```
public Node item(int index)
```

参数说明

index：表示当前节点的索引。

（1）创建一个 XML 文档用于 DOM 解析。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<book:books xmlns:book="http://www.mingrisoft.com"
<!--声明命名空间-->
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.mingrisoft.com books.xsd">
<!--声明命名空间的存储地址-->
        <book:book>
                <book:name>《C#从入门到精通(第 2 版)》</book:name>
                <book:publisher>清华大学出版社</book:publisher>
                <book:company>明日科技</book:company>
                <book:author>王小科</book:author>
                <book:ISBN>9787302226628</book:ISBN>
                <book:price unit="yuan" unitType="RMB">69.80</book:price>
                <book:url>http://www.mingribook.com/bookinfo.php?id=227</book:url>
        </book:book>
        <book:book>
                <book:name>《JavaScript 开发技术大全》</book:name>
                <book:publisher>人民邮电出版社</book:publisher>
                <book:company>明日科技</book:company>
                <book:author>梁冰</book:author>
                <book:ISBN>9787115179708</book:ISBN>
                <book:price unit="yuan" unitType="RMB">65.00</book:price>
                <book:url>http://www.mingribook.com/bookinfo.php?id=138</book:url>
        </book:book>
```

```
</book:books>
```

（2）创建 AttriburteDOMParserFile 类，在该类中创建 parseReadFile()方法用于读取 XML 文件，同时返回一个 Document。代码如下：

```
public class AttriburteDOMParserFile {                                    //读取 XML 文件
        public Document parseReadFile(String path)
                        throws ParserConfigurationException, SAXException, IOException {
                        DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory
                                .newInstance();                                //创建 DocumentBuilderFactory 实例
                        DocumentBuilder dombuilder = documentBuilderFactory
                                .newDocumentBuilder();                         //创建 DocumentBuilder 实例
                        File file = new File(path);                            //获取 XML 文件
                return dombuilder.parse(file);                                 //解析 XML 文件将结果返回
        }
}
```

（3）建立一个 List 变量，用于保存 XML 文件的元素和内容。代码如下：

```
private List<String> elementList = new ArrayList<String>();
```

（4）创建一个 getElementName()方法，在该方法内部读取 XML 文件元素、属性和属性值，然后拼成一个字符串，再保存到 List 变量里。代码如下：

```
public void getElementName(Node parentNode) {
        if (parentNode.hasChildNodes()) {
                NodeList nodeList = parentNode.getChildNodes();
                for (int i = 0; i < nodeList.getLength(); i++) {
                        Node node = nodeList.item(i);
                        if (node.hasChildNodes()) {
                                getElementName(node);
                                //获取属性列表
                                NamedNodeMap namedNodeMap = node.getAttributes();
                                for (int j = 0; j < namedNodeMap.getLength(); j++) {
                                        Node node2 = namedNodeMap.item(j);
                                        //获取属性值
                                        elementList.add(node.getNodeName() + " = "+ node2.getNodeName() + " > "+ node2.getNodeValue());
                                }
                        }
                }
        }
}
```

（5）创建 getElementList()方法，返回保存在 List 变量中的结果。代码如下：

```
public List<String> getElementList() {
        return this.elementList;                        //将结果保存到 List 中
}
```

（6）创建 main()方法，在其内部调用 AttriburteDOMParserFile 的 parseReadFile()方法、getElementName()方法和 getElementList()方法，然后把结果输出到控制台。代码如下：

```
public static void main(String[] arg) {
                AttriburteDOMParserFile parserFile = new AttriburteDOMParserFile();
                //创建 AttriburteDOMParserFile 的实例
                String path = "xmldemo/books.xml";
                try {
                        Document document = parserFile.parseReadFile(path);
                        parserFile.getElementName(document);
                        List<String> list = parserFile.getElementList();
                        System.out.println("属性名称和属性值");
                        System.out.println(list);
                } catch (ParserConfigurationException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (SAXException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }
```

秘笈心法

心法领悟 035：单一获取属性的另一方式。

在本实例中是根据 Node 的 getNodeValue()方法获取属性的值，如果只针对属性而言，使用 getTextContent()方法也可以获取属性值，getTextContent()和 getNodeValue()方法的内容是一样的。

实例 036　使用 SAX 验证 DTD

光盘位置：光盘\MR\01\036

高级

实用指数：★★★★

实例说明

使用 SAX 也可以验证 XML 是否符合 DTD 的规范。首先 XML 要引用 DTD 文件，然后通过 SAX 对当前的 XML 进行解析，解析时需要设置 factory.setValidating(true)。本实例要解析的是 XML 和 DTD，为了能够看到验证的效果，把 XML 元素中的 price 元素改成 prices，解析 XML 时就会出现如图 1.36 所示的错误信息。

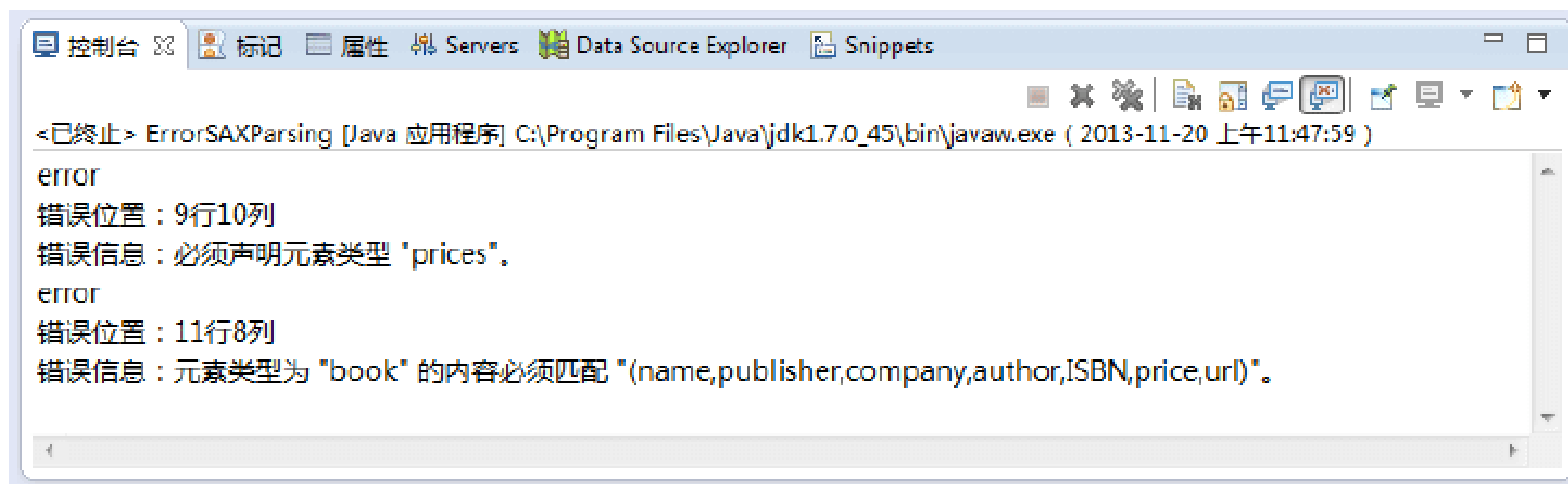


图 1.36　使用 SAX 组件验证 DTD

关键技术

在使用 SAX 解析 XML 时，要先创建 ErrorSAXParsing 类，继承 DefaultHandler 类，在 ErrorSAXParsing()上重写 warning()、error()和 fatalError()方法（3 个错误处理分别表示 3 种不同程度的错误）。验证 XML 是否符合 DTD 规范需要覆盖 error()方法。其语法如下：

```
public void error(SAXParseException exception) throws SAXException
```

参数说明

exception：解析 XML 时发生的异常处理。

（1）创建一个 DTD 文档用来定义 XML 文档的格式。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明文档版本与字符编码方式-->
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!--定义元素 book 及使用规则-->
<!ELEMENT name        (#PCDATA)>
<!ELEMENT publisher   (#PCDATA)>
<!ELEMENT company     (#PCDATA)>
<!ELEMENT author      (#PCDATA)>
<!ELEMENT ISBN        (#PCDATA)>
<!ELEMENT price       (#PCDATA)>
<!ELEMENT url         (#PCDATA)>
<!ATTLIST price unit CDATA "RMB" >
```

（2）根据 XSD 文档的结构定义 XML 文档。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--声明 XML 文档版本与字符编码方式-->
<!DOCTYPE book SYSTEM "books.dtd">
<!--引入外部 DTD 文件-->
<book>
```

```
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price>69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

（3）创建 ErrorSAXParsing 类，继承 DefaultHandler 类，实现 error()方法、warning()方法和 fatalError()方法，在 3 个方法中向控制台输出错误位置和错误信息。Error()方法代码如下：

```
public void error(SAXParseException exception) throws SAXException {
        System.out.println("error");
        System.out.println("错误位置：" + exception.getLineNumber() + "行" + exception.getColumnNumber() + "列");
        System.out.println("错误信息：" + exception.getMessage());
}
```

（4）创建 parseReadFile()方法，把 ErrorSAXParsing 实例传入解析器，实现 XML 的解析。代码如下：

```
public void parseReadFile(String pathname) {
                SAXParser parser;
                SAXParserFactory factory = SAXParserFactory.newInstance();          //创建 SAXParserFactory 实例 factory
                try {
                        factory.setValidating(true);
                        factory.setNamespaceAware(true);
                        parser = factory.newSAXParser();                            //获取 SAXParser 实例
                        File file = new File(pathname);                             //获取文件
                        parser.parse(file, this);                                   //解析 XML 文件
                } catch (ParserConfigurationException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (SAXException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }
```

（5）创建 main()方法，在该方法中使用 ErrorSAXParsing 解析 books.xml 文件，输出错误信息到控制台。代码如下：

```
public static void main(String[] arg) {
                String pathname = "xmldemo/books.xml";
                ErrorSAXParsing elementSAXParsing = new ErrorSAXParsing();     //创建 ErrorSAXParsing 的实例
        elementSAXParsing.parseReadFile(pathname);  //解析 XML 文件
        }
```

心法领悟 036：SAXParseException 类的方法介绍。

使用 SAXParseException 类的 getLineNumber()方法、getColumnNumber()方法和 getMessage()方法可以获取错误发生的行号、列号以及具体的错误内容。语法如下：

```
//行号
public int getLineNumber ()
//列号
public int getColumnNumber ()
//错误内容
public String getMessage ()
```



实例 037　使用 dom4j 解析 XML 文件

光盘位置：光盘\MR\01\037

高级

实用指数：★★★★

实例说明

dom4j 是一种解析 XML 文档的开放源代码 XML 框架。应用 dom4j 框架解析 XML 文件，需要 jaxen.jar 包的支持。本实例实现的是使用 dom4j 向 XML 文件中写入数据，并将用户输入的信息保存到 XML 文件中，运行结果如图 1.37 所示。



图 1.37　使用 dom4j 解析 XML 文件

关键技术

本实例实现的是应用 dom4j 向 XML 文件中写入数据，以下是涉及的类及方法。

（1）SAXReader 类

该类用于解析 XML 文档。

（2）Document 类

该类是一个文档实例。通过该类的 addelement()方法可实现向 XML 文档中添加元素，并可获取封装文档子元素的 Element 对象。例如，创建根元素 catalog。代码如下：

```
Element catament = document.addelement("catalog");
```

（3）Element 类

该类封装了文档中的元素信息。该类中的常用方法有：

☑ addcomment()方法：该方法用于向XML文档中添加注释。它有一个String类型的参数，注释内容为参数值。例如，下面的代码向catalog元素添加注释“a xml catalog”。

```
catament.addcomment("a xml catalog");
```

☑ addelement()方法：该方法用于向XML文件中添加子元素。它有一个String类型的参数，添加的子元素名称为参数内容。例如，下面的代码向XML文件中增加journal元素。

```
Element journalelement = catalogelement.addelement("journal");
```

☑ settext()方法：该方法用于设置元素的文本。它有一个String类型的参数，文本内容为参数值。例如，下面的代码为journal元素设置文本：

```
journalelement.settext("hello word");
```

☑ addAttribute()方法：该方法用于向元素中添加属性。它有两个String类型的参数。语法如下：

```
addAttribute(String name ,String value)
```

参数说明

❶ name：指定属性名称。

❷ value：指定属性值。

（1）本实例实现的是将用户输入的论坛信息保存到 XML 文件中。XML 文件中每条论坛信息对应着一个 leave 根元素，该元素还包含着多项子元素。XML 文件的结构如下：

```
<?xml version="1.0" encoding="GBK"?>
<!--声明 XML 文档版本与字符编码方式-->
<leave>
        <date type="Gerver">2009 年 3 月 25 日  星期三</date>
        <name>mr</name>
```

```
        <title>《Java Web 范例宝典》</title>
        <content>出版了！</content>
</leave>
```

（2）根据 XML 文件定义的格式，创建 Java Bean 类 Leave，该类中所包含属性与 XML 文件中的各子元素相对应，并包含了各属性的 setXXX()与 getXXX()方法。具体代码如下：

```
public class Leave {
   private String name;
   private String date;
   private String title;
   private String content;
public String getName() {
 return name;
}
public void setName(String name) {
 this.name = name;
}
                    ……//省略了其他属性的 setXXX()与 getXXX()方法
}
```

（3）本实例将用户输入留言信息的时间保存到 XML 文件中，留言时间并不需要用户输入，系统将获取当前时间，保存到 XML 文件中。具体代码如下：

```
public class GetTime {
        public static String currentlyTime() {
                Date date = new Date();                          //创建 Date 对象
                DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.FULL);
                return dateFormat.format(date);                  //对系统时间格式化输出
        }
}
```

（4）创建向 XML 文件中添加元素及元素值的方法 addXmlNode()，该方法有两个参数，分别用于指定 XML 文件的地址以及 Leave 对象。具体代码如下：

```
public boolean addXmlNode(String fileName,Leave leave){
        boolean rtn = false;
        SAXReader reader = new SAXReader();                          //创建 SAXReader 对象，解析 XML 文档
        try{
            Document document = reader.read(new File(fileName));      //创建文档对象
            List list = document.selectNodes("/leave");               //获取 leave 元素中的子元素
            Iterator itr = list.iterator();
            Element journalElement = (Element)itr.next();
            Element dateElement = journalElement.addElement("date");  //向 XML 文件中添加元素
            dateElement.setText(leave.getDate());                     //设置元素值
            dateElement.addAttribute("type", "Gerver");               //设置元素属性
            Element nameElement = journalElement.addElement("name");  //向元素中添加子元素
            nameElement.setText(leave.getName());                     //设置 name 元素值
            Element titleElement = journalElement.addElement("title"); //向元素中添加 title 子元素
            titleElement.setText(leave.getTitle());                   //设置 title 子元素值
            Element contentElement = journalElement.addElement("content"); //添加 content 子元素
            contentElement.setText(leave.getContent());               //设置 content 子元素值
            XMLWriter output;
            OutputFormat format = OutputFormat.createCompactFormat(); //创建 OutputFormat 对象
            format.setEncoding("GBK");                                //设置写入流编码
            output = new XMLWriter(new FileWriter(fileName),format);
            output.write(document);                                   //向流写入数据
            output.close();                                           //关闭流
            rtn = true;
        }
        catch (Exception e) {
                e.printStackTrace();
        }
        return rtn;
}
```

（5）当用户输入“帖子主题”、“帖子内容”后，单击“提交”按钮，系统将提交 URL 地址 AddLeaveServlet。在该 Servlet 中将调用 addXmlNode()方法，实现将数据写入 XML 文件中。Servlet 中的代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
```

```
        String filePath = request.getRealPath("/myxml/forum.xml");          //获取 XML 文件保存地址
        CreateXMl createXml = new CreateXMl();                              //创建保存有向 XML 文件中添加数据的类对象
        Leave leave = new Leave();                                          //创建 Leave 类对象
        leave.setName("mr");                                                //设置 Leave 类对象默认用户名
        leave.setTitle(request.getParameter("title"));                      //设置 Leave 类对象帖子主题
        leave.setContent(request.getParameter("content"));                  //设置 Leave 类对象帖子内容
        leave.setDate(GetTime.currentlyTime());                             //设置 Leave 类对象发表时间
        String message = "帖子发表成功";
        boolean bool = createXml.addXmlNode(filePath,leave);                //调用添加 XML 文件方法
        if(bool == false)
                message = "帖子发表失败";
        request.setAttribute("message",message);                            //将信息保存在 request 对象中
        RequestDispatcher requestDispatcher =
                request.getRequestDispatcher("index.jsp");                  //设置转发地址
        requestDispatcher.forward(request, response);
}
```

秘笈心法

心法领悟 037：dom4j 的扩展应用。

根据本实例，读者可以做到：

使用dom4j实现向XML文件中添加数据。

使用dom4j实现解析XML文件中的数据。

第 2 章

发送与接收邮件

- ⏭ 配置邮件服务器
- ⏭ 应用 JavaMail 组件发送邮件
- ⏭ 应用 JavaMail 组件接收邮件
- ⏭ 应用 Apache commons-email 组件发送邮件
- ⏭ 应用 Spring 的 E-mail 抽象层发送邮件

2.1　配置邮件服务器

在互联网上发送和接收电子邮件之前，必须配置 SMTP 服务器和 POP3 服务器。下面分别介绍在 Windows Server 2003 系统中配置邮件服务器、Apache 的开源邮件服务器（Apache James Server）和 Magic Winmail 邮件服务器。

注意： 在实际开发应用中，只需要配置其中一种邮件服务器即可，在此只是为了演示不同邮件服务器的配置方法。如果要配置多个不同类型的邮件服务器，需要注意的是，指定的 SMTP 或 POP3 的端口号不能相同，以免发生冲突，因为需要使用 SMTP 和 POP3 端口协议来发送和接收邮件。

实例 038	在 Windows Server 2003 系统下安装和配置邮件服务器 光盘位置：光盘\MR\02\038	初级 实用指数：★★★★

实例说明

本实例以 Windows Server 2003 操作系统为例，演示如何安装和配置 SMTP 服务器和 POP3 服务器。

关键技术

发送邮件服务器使用的是邮件发送协议，现在常用的是 SMTP 协议。接收邮件服务器使用邮件接收协议，常用的有 POP3 协议和 IMAP 协议。所以，在使用 Windows Server 2003 作为邮件服务器时，首先需要安装和配置 SMTP 服务器和 POP3 服务器。

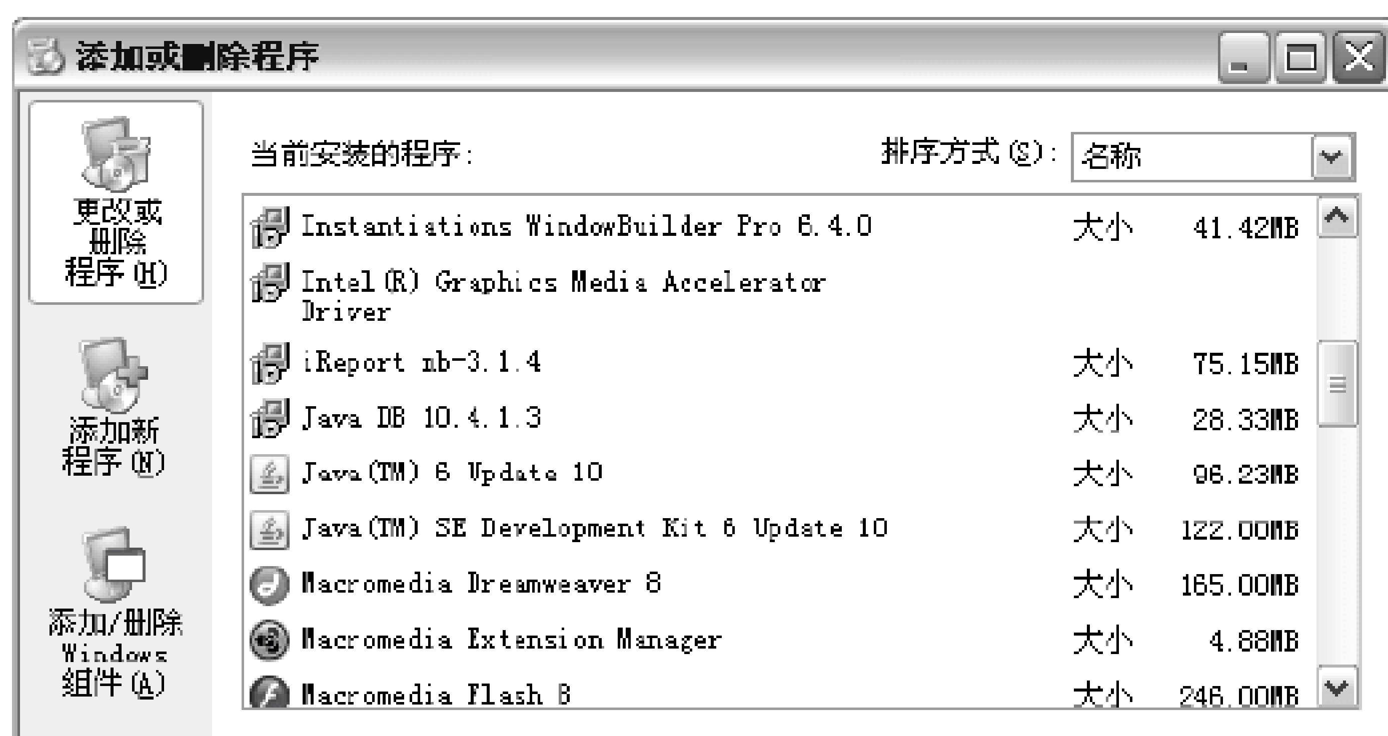


图 2.1　“添加或删除程序”窗口

（1）打开“控制面板”，选择“添加或删除程序”选项，打开如图 2.1 所示的“添加或删除程序”窗口。

（2）单击“添加/删除 Windows 组件”按钮，打开“Windows 组件向导”对话框。在该对话框的“组件”列表框中选中“电子邮件服务”和“应用程序服务器”复选框，如图 2.2 和图 2.3 所示。

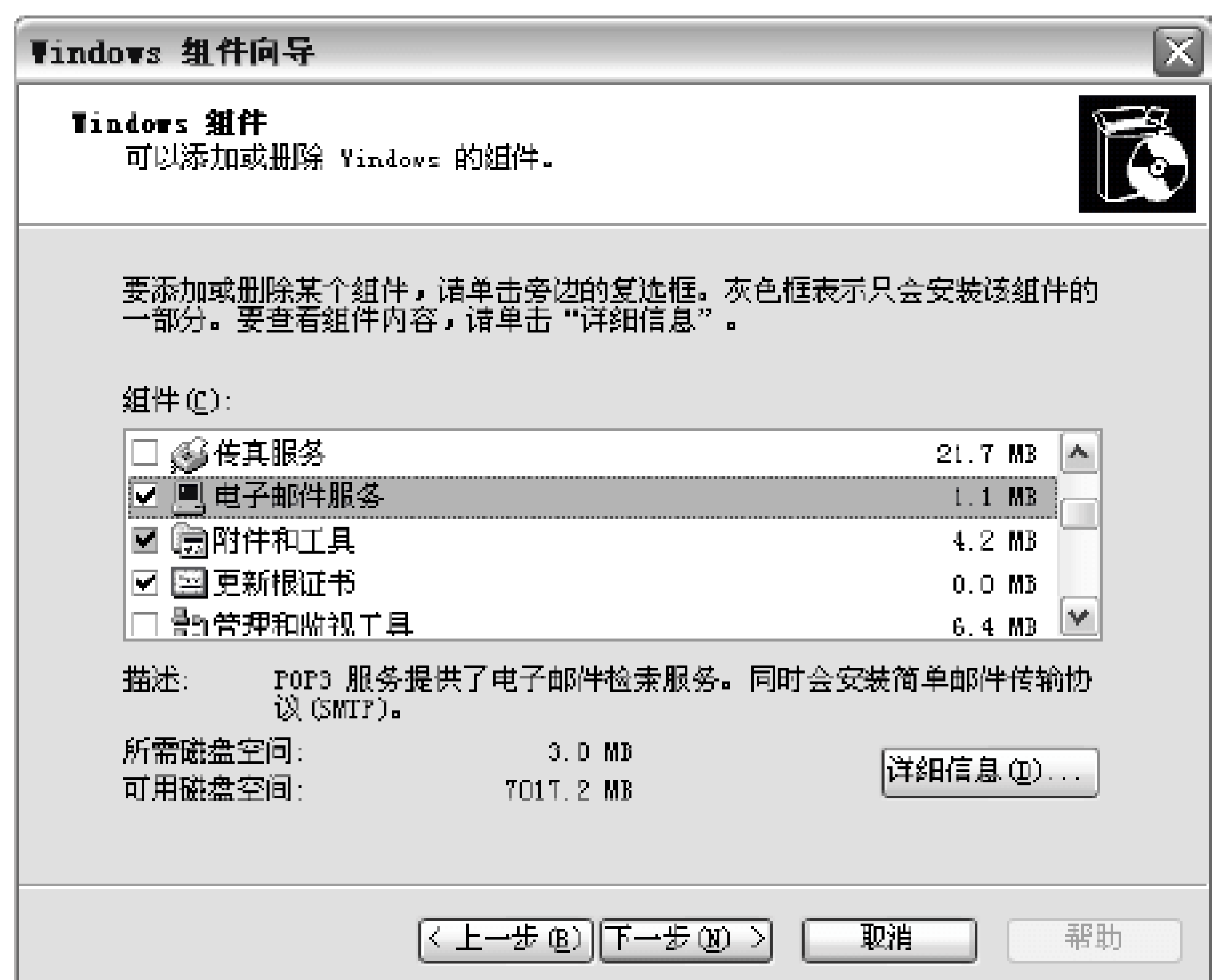


图 2.2　选中“电子邮件服务”复选框

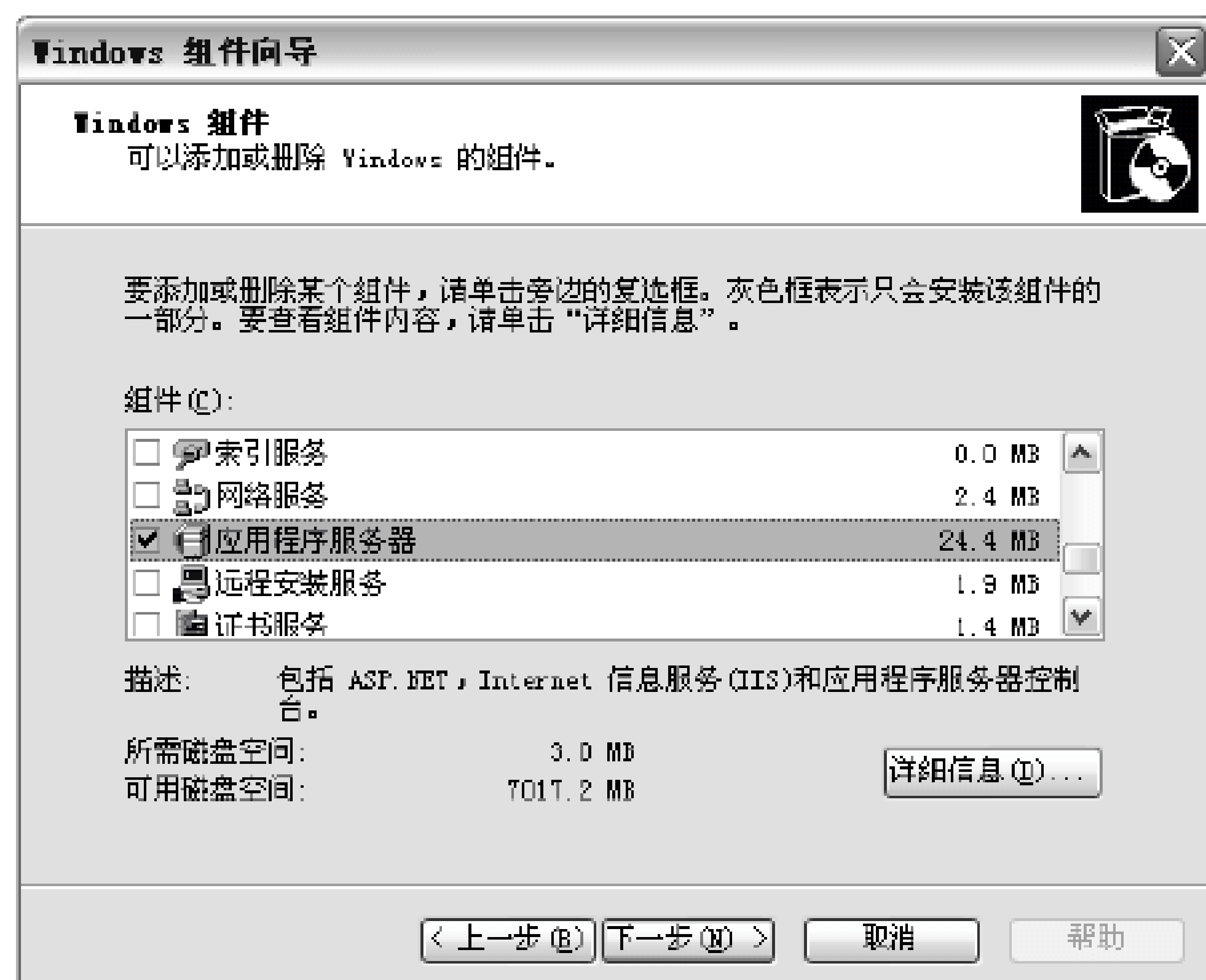


图 2.3　选中“应用程序服务器”复选框

（3）在图 2.3 中单击“详细信息”按钮，在弹出的“应用程序服务器”对话框中选中“Internet 信息服务（IIS）”

复选框，如图 2.4 所示。

（4）在图 2.4 中单击“详细信息”按钮，在弹出的“Internet 信息服务（IIS）”对话框中选中 SMTP Service 复选框，如图 2.5 所示。

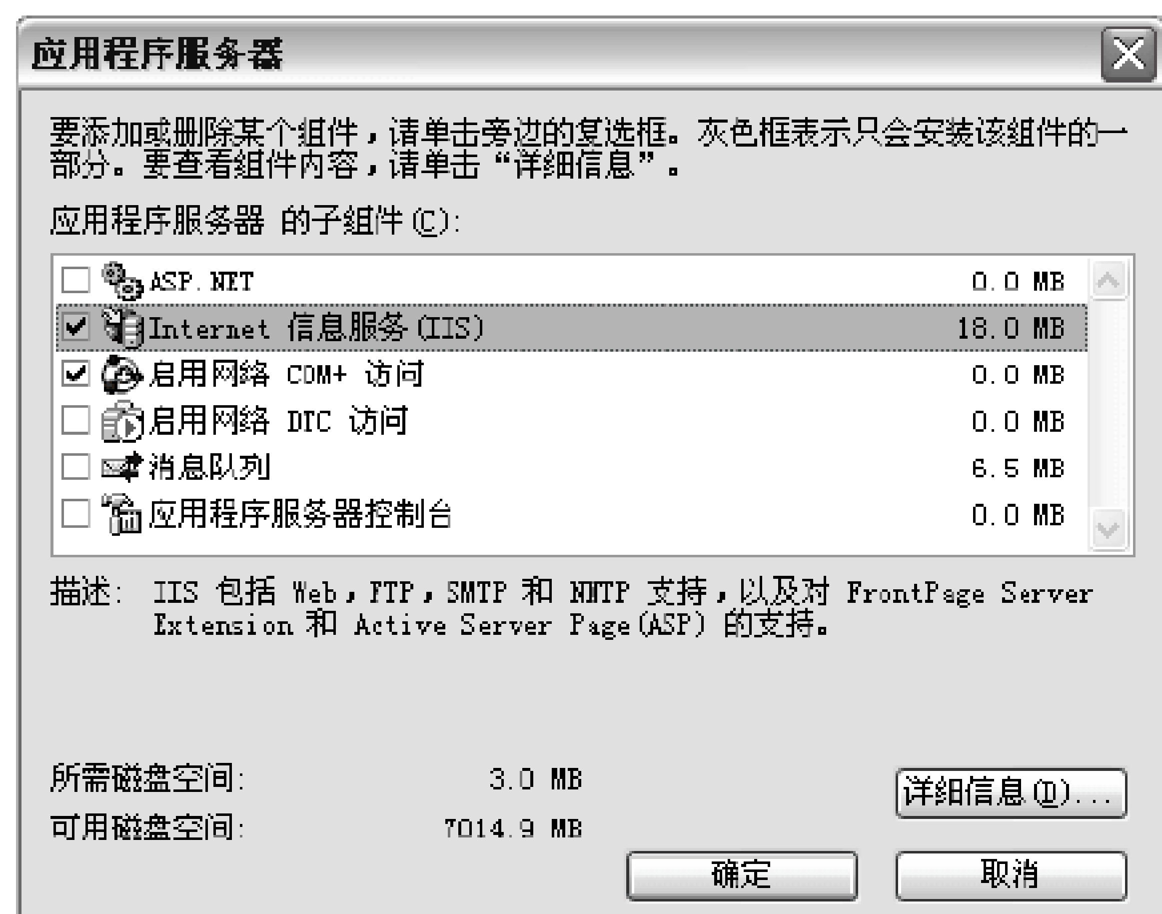


图 2.4　选中“Internet 信息服务（IIS）”复选框

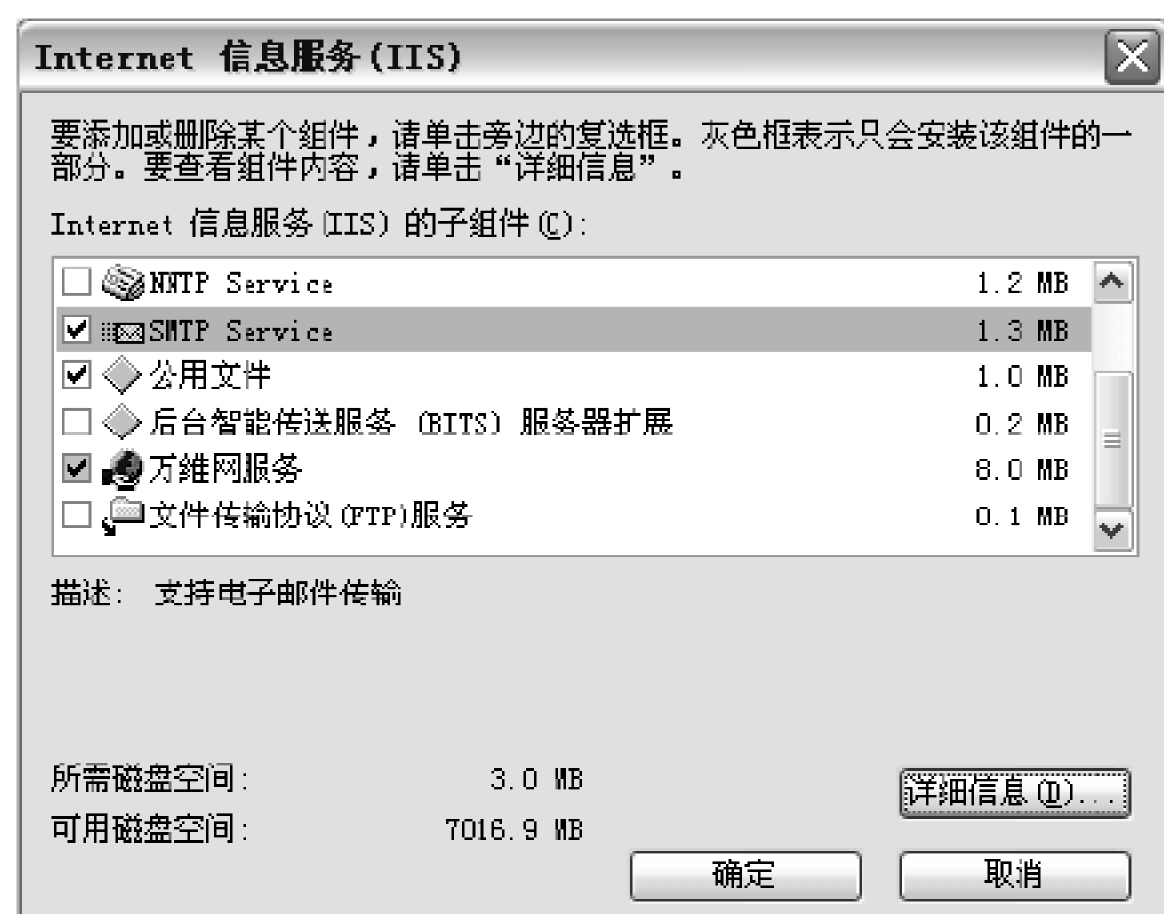


图 2.5　选中 SMTP Service 复选框

（5）单击“确定”按钮，返回“应用程序服务器”对话框。单击“确定”按钮，返回“Windows 组件向导”对话框。单击“下一步”按钮，将开始安装 SMTP 和 POP3 服务器。

（6）邮件服务器安装完成后，运行“控制面板”/“管理工具”/“POP3 服务”程序，弹出“POP3 服务”窗口。在左侧的列表框中选择 POP3 服务器名（这里为 WANGGH）节点，右击，在弹出的快捷菜单中选择“所有任务”子菜单中的相应命令，可以实现 POP3 服务器的启动、停止或暂停；选择“新建”/“域”命令，如图 2.6 所示，将弹出“添加域”对话框。

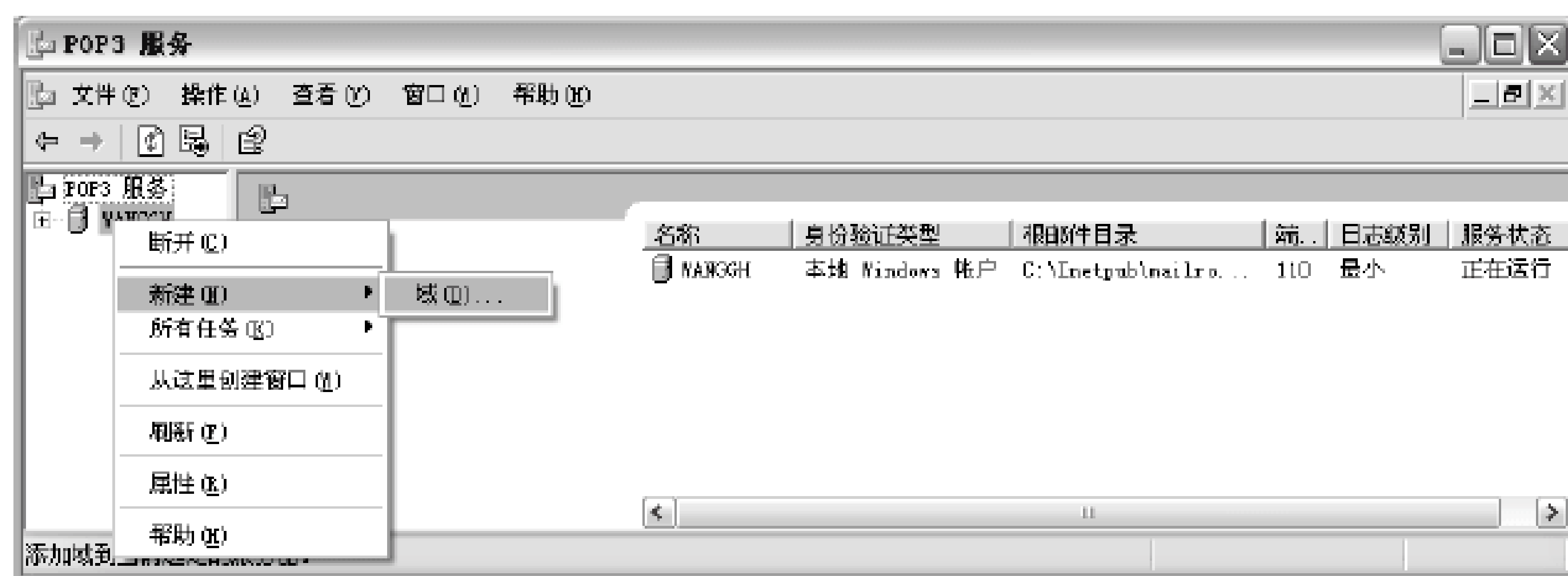


图 2.6　选择“新建”/“域”命令

（7）在“添加域”对话框的“域名”文本框中输入域名 yahoo.com，单击“确定”按钮，即可创建一个新的域。此时在 SMTP 中将自动创建一个名为 yahoo.com 的域。创建新域后，在“POP3 服务”窗口右侧将显示该域的基本信息。在域名称 yahoo.com 上右击，在弹出的快捷菜单中选择“新建”/“邮箱”命令，如图 2.7 所示，将打开“添加邮箱”对话框。

（8）在“添加邮箱”对话框的“邮箱名”文本框中输入邮箱名称 wgh717，在“密码”文本框中输入邮箱密码，在“确认密码”文本框中确认密码，如图 2.8 所示，然后单击“确定”按钮，完成邮箱 wgh717@yahoo.com 的创建。按照该步骤再创建一个名为 test123@yahoo.com 的邮箱。



图 2.7　新建邮箱



图 2.8　添加邮箱

说明：在同一个域中，可以创建多个邮箱。

秘笈心法

心法领悟 038：配置 SMTP 和 POP3 的端口号。

在配置 SMTP 和 POP3 服务器时，需要注意端口号是否被占用，如果被占用，则需要将端口号改成其他的，这样在发送和接收邮件时才会成功。

实例 039　配置开源邮件服务器 Apache James Server

光盘位置：光盘\MR\02\039

中级

实用指数：★★★★

实例说明

James Server 是 Apache 组织开发的一个开源的邮件服务器，也是该组织开发的子项目之一。其优点突出，主要体现在性能稳定、可配置性强，并且是开源项目，所有源代码不存在版权问题。因此，在项目中的应用非常广泛。可以访问 http://james.apache.org 网站下载 James Server。本实例使用的版本为 James Server 2.3.2，该版本不仅支持基本的 SMTP 和 POP3 协议，还支持 NNTP 协议（NNTP 用于客户端从新闻服务器存储和获取消息）。

下面介绍如何配置 Apache James Server 邮件服务器。安装完 James Server 之后，执行其安装目录下 bin 文件夹中的 run.bat 文件，即可启动 James Server 邮件服务器，如图 2.9 所示。

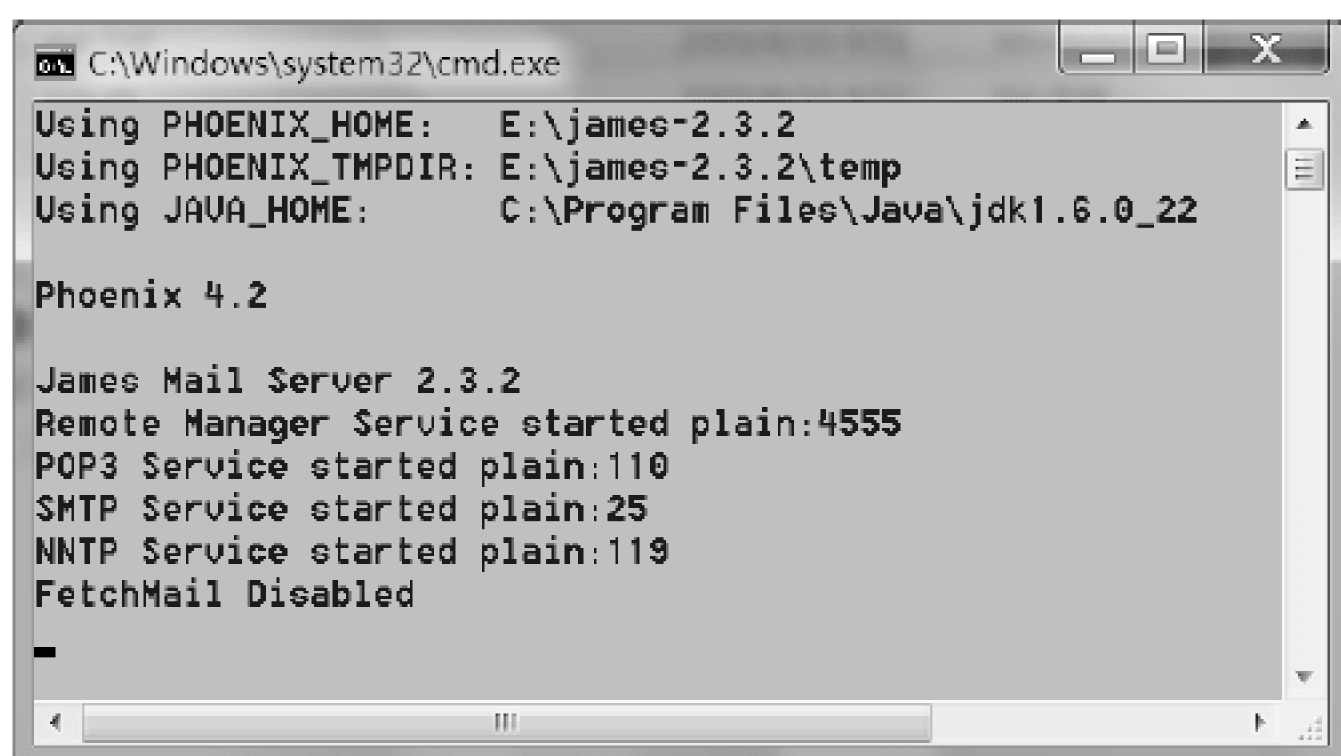


图 2.9　James Server 邮件服务器的启动界面

注意：在启动 James Server 之前，应确保 POP3、SMTP 和 NNTP 服务的端口号没有被占用，如果被占用，将无法启动。

关键技术

James Server 完全是基于 Java 开发的邮件服务器，在配置该邮件服务器之前，需要有 JVM 的支持。因此，首先需要安装 JDK，然后配置 JAVA_HOME（因为 James Server 在启动时，需要根据 JAVA_HOME 的环境变量来查找 JDK 的安装位置）。

（1）第一次启动 James Server 后，会在其安装目录下生成一个 apps 文件夹。进入 apps\james\SAR-INF 目录，打开 config.xml 文件，该文件用于配置 James Server。

（2）在 config.xml 文件中，找到<postmaster>Postmaster@localhost</postmaster>，把此项改为<postmaster>Postmaster@unitname</postmaster>。

（3）找到<servername>localhost</servername>，把此项改为<servername>unitname</servername>。修改这两项的目的是将默认的 localhost 改为机器名，使其他机器也能访问邮件系统。当然，前提是在局域网中没有与服务器重名的机器。

（4）将以下内容注释掉。

```
<mailet match="RemoteAddrNotInNetwork=127.0.0.1" class="ToProcessor">
        <processor> relay-denied </processor>
        <notice>550 - Requested action not taken: relaying denied</notice>
</mailet>
```

（5）将以下内容的注释去掉，使其生效。

```
<authRequired>true</authRequired>
```

（6）查找所有的 myMailServer，替换为自己的邮箱域名。例如：

```
<helloName autodetect="false">myMailServer</helloName>
```

注意：将 myMailServer 修改为自己的邮箱域名，如 test.com。

（7）修改<authRequired>的值为 true，打开 SMTP 的认证。关键代码如下：

```
<authRequired>true</authRequired>
```

（8）修改 root 口令。关键代码如下：

```
<account login="root" password="111"/>
```

（9）在命令行中通过 telnet 命令登录 James Server。关键代码如下：

```
telnet localhost 4555
```

（10）连接到主机之后，需要输入登录 id 和密码（也就是 root 用户名和密码），然后输入 help 命令，查看所有的配置命令，如图 2.10 所示。

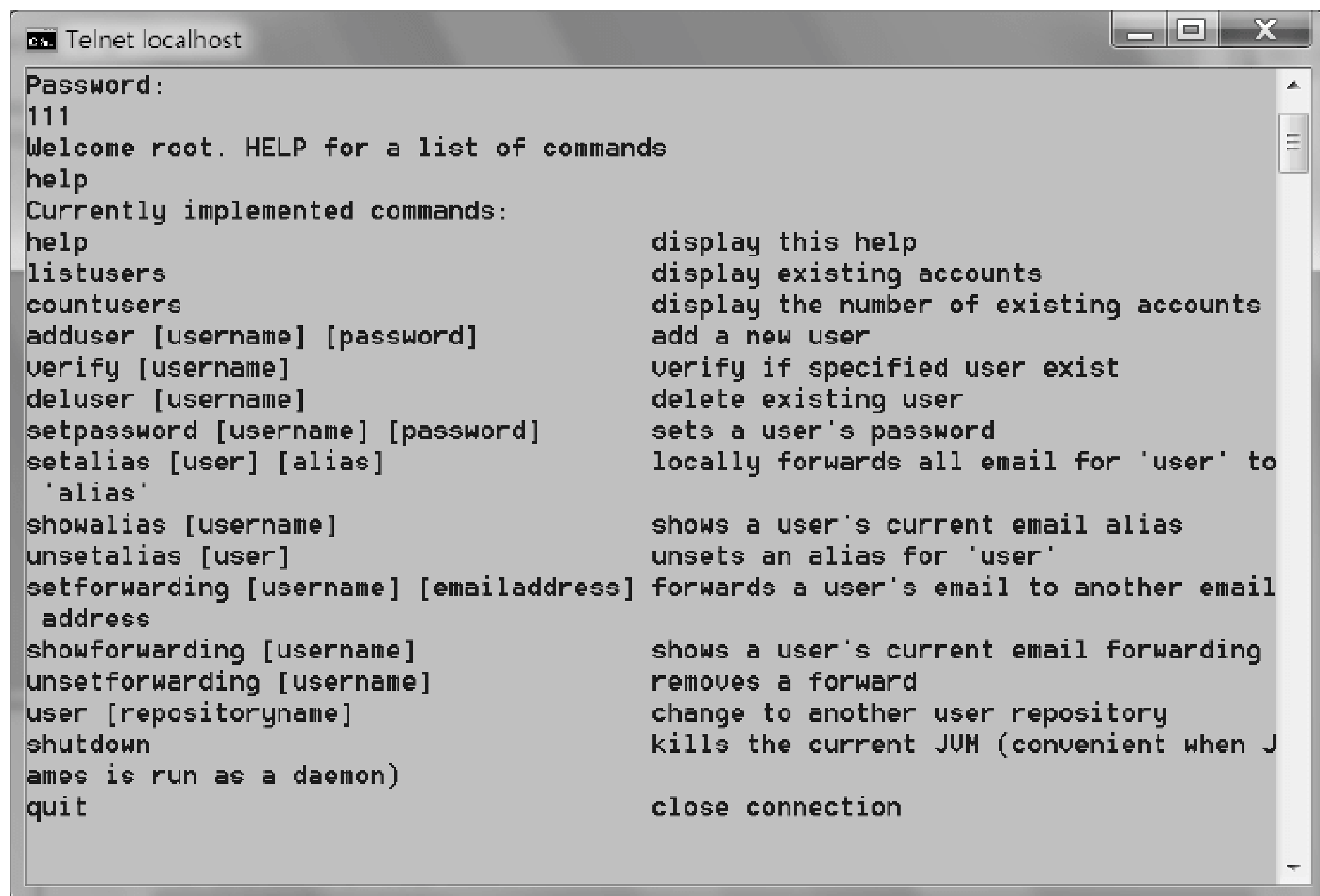


图 2.10　登录 James Server 后的效果

秘笈心法

心法领悟 039：添加邮箱用户。

通过 addusers [username] [password]命令即可添加邮箱用户。例如：

```
adduser test 111111
```

添加邮箱用户之后，假如配置的邮箱域名为 abc.com，那么就可以通过 Outlook 访问邮箱了，邮箱地址为 test@abc.com，密码为 111111。

实例 040　安装和配置 Magic Winmail 邮件服务器　高级

光盘位置：光盘\MR\02\040　实用指数：★★★★

Winmail 是一款易用型全功能邮件服务器软件，功能比较完善，但它并不是免费的。可以访问 http://www.magicwinmail.net 网站下载该邮件服务器。Winmail 支持标准的 SMTP、POP3、IMAP 邮件协议，而且支持 WebMail，也就是用户可以通过 Web 网页访问邮件服务器，大大方便了邮件服务器的维护、管理。下面介绍一下 Winmail

邮件服务器的安装及配置。安装 Winmail 之后，在 Web 网页中访问邮件服务器的效果如图 2.11 所示。



图 2.11　Winmail Server 邮件服务器运行效果

关键技术

如果操作系统中已经安装了其他邮件服务器，在安装配置 Winmail 时，需要注意 SMTP、POP3 和 IMAP 邮件协议所用端口的配置。SMTP 的默认端口号为 25，POP3 的默认端口号为 110，IMAP 的默认端口号为 143。如果端口号发生冲突，可以对这几个端口进行修改。

设计过程

（1）双击运行 Winmail 的安装文件，弹出如图 2.12 所示的欢迎界面。

（2）单击 Next 按钮，进入是否同意安装协议的界面，选中 I accept the agreement 单选按钮，如图 2.13 所示。



图 2.12　安装 Winmail Mail Server 欢迎界面

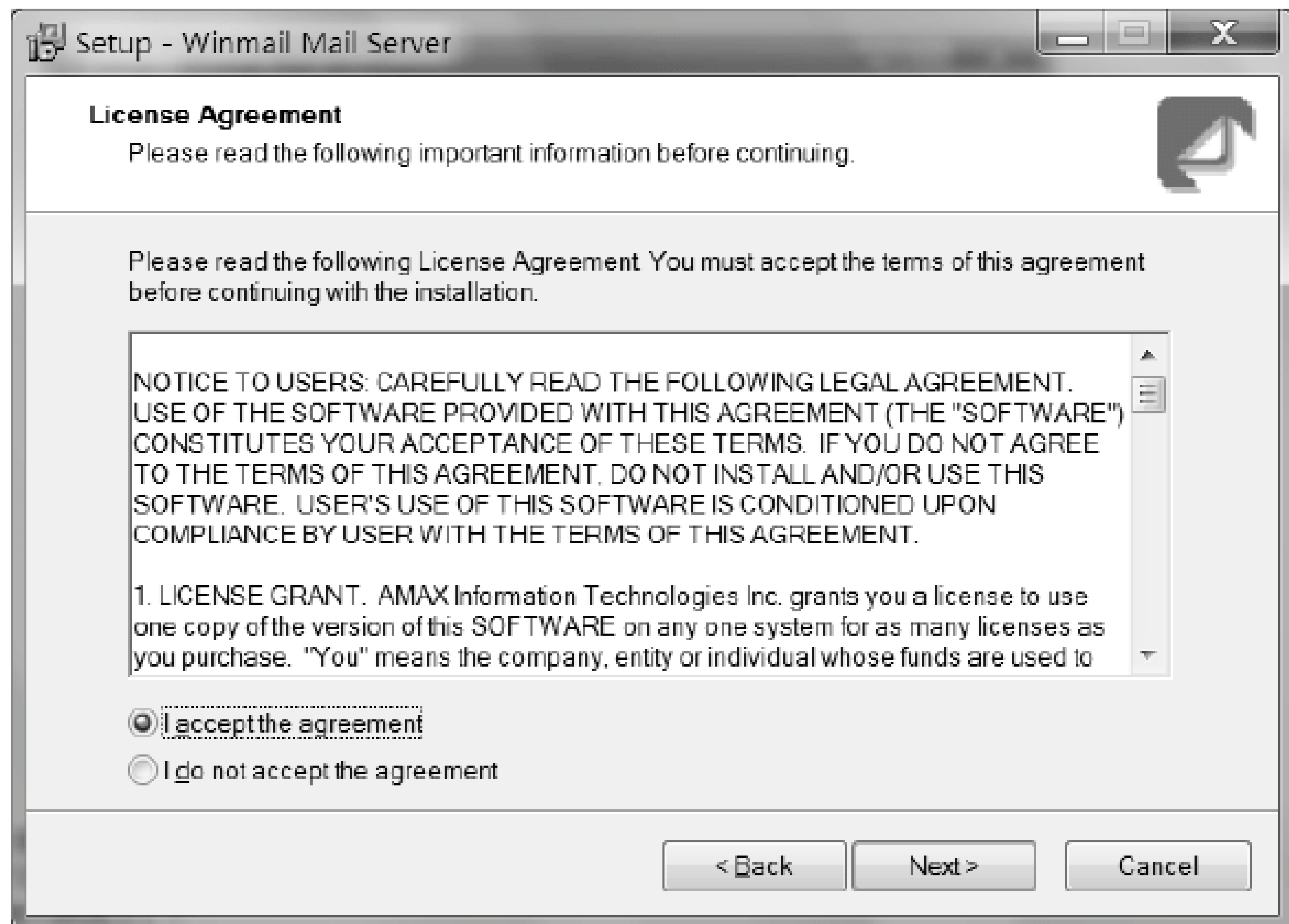


图 2.13　是否同意安装协议的界面

（3）单击 Next 按钮，进入填写个人信息的界面；继续单击 Next 按钮，进入选择安装路径的界面（如图 2.14 所示），单击 Browse 按钮可修改安装路径。

（4）单击 Next 按钮，在弹出的界面中选择要安装的组件，如图 2.15 所示。在此保持默认设置，直接单击 Next 按钮，开始执行安装程序。

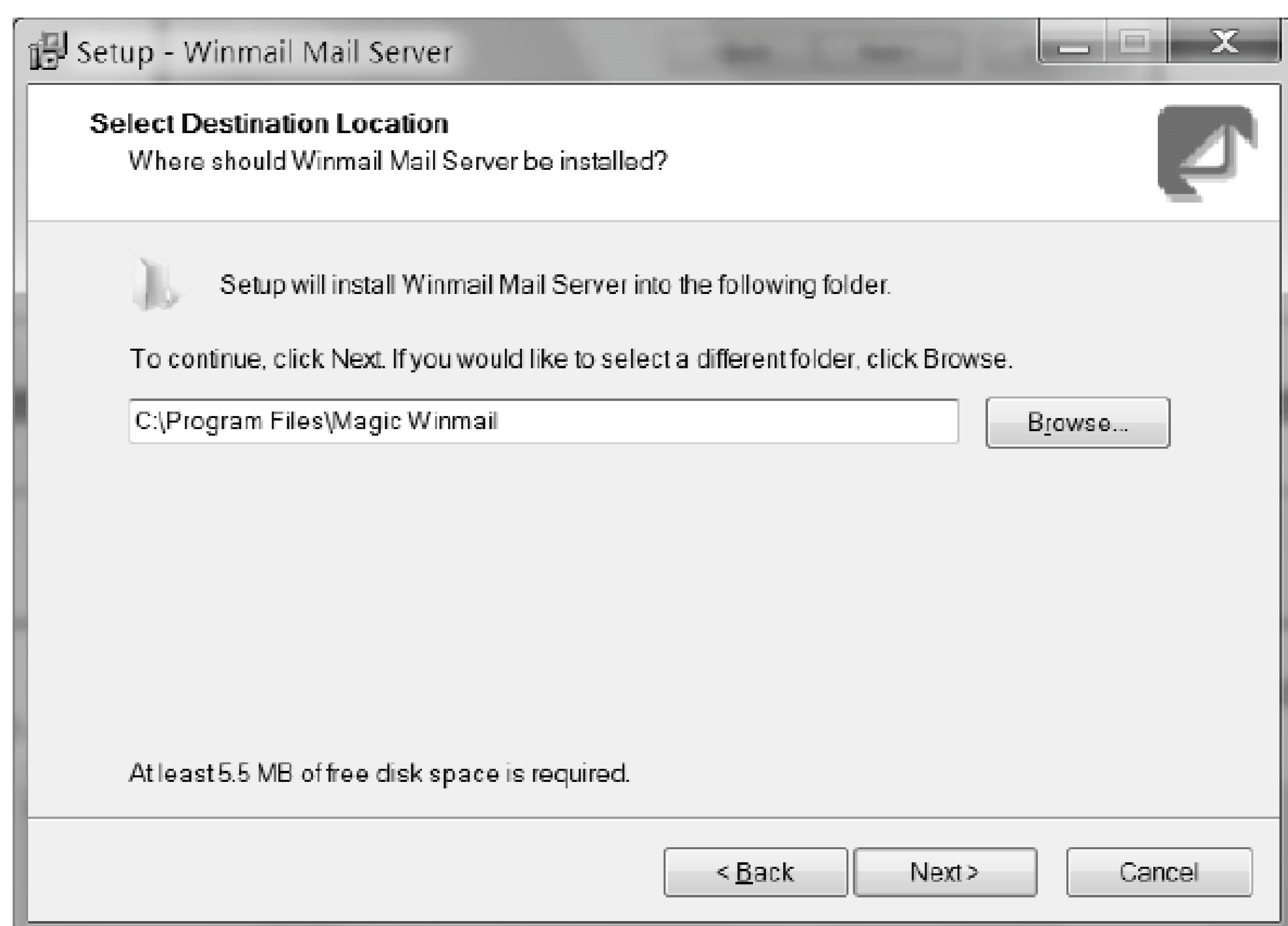


图 2.14　选择要安装路径

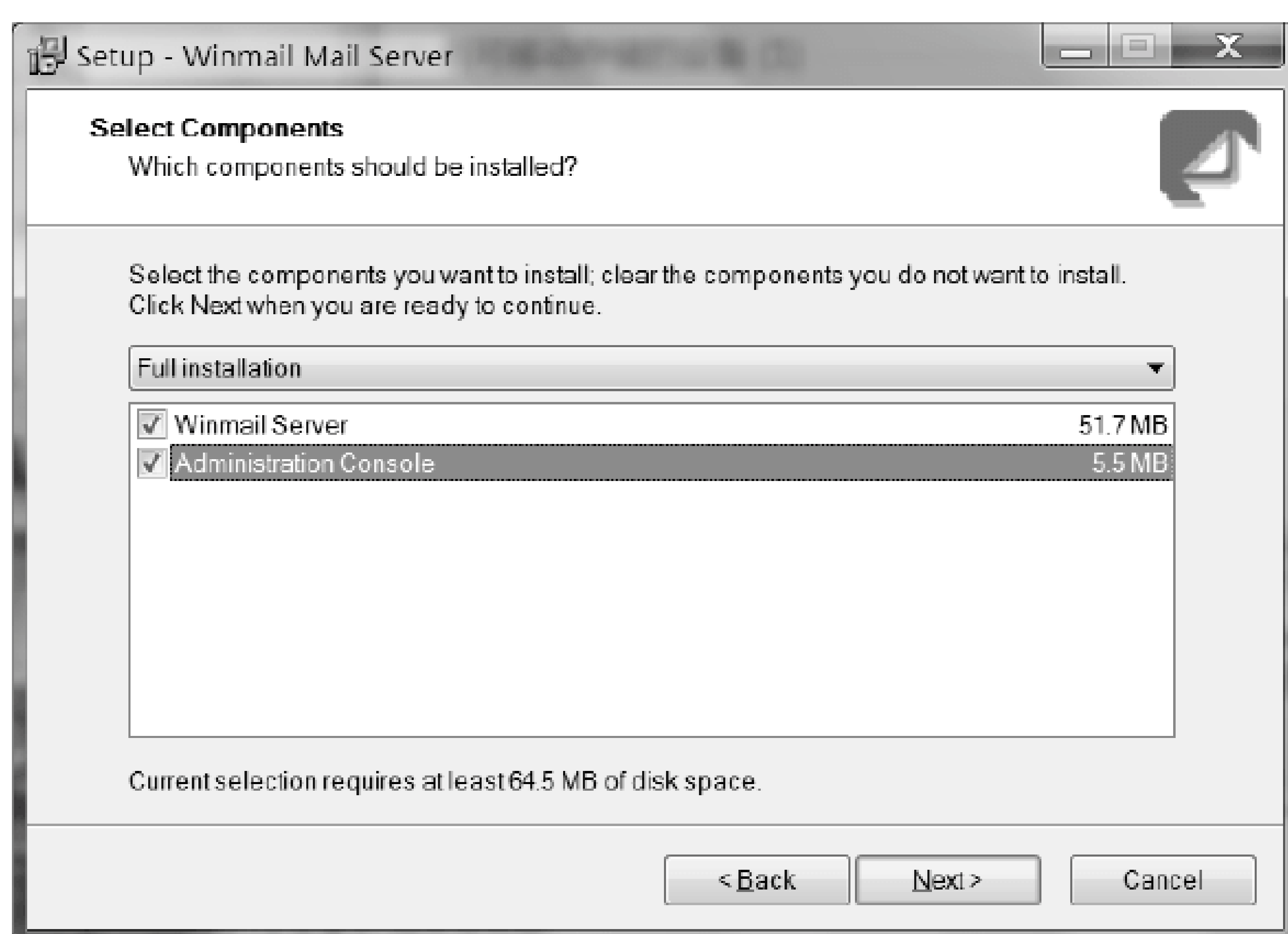


图 2.15　选择要安装的组件

（5）完成安装后，文件列表如图 2.16 所示。

（6）双击 Magic Winmail Adminstration Console 文件，进入 Winmail 邮件服务器的管理界面，如图 2.17 所示。在这个管理器中，可以对相关邮件协议进行配置，对域、用户等进行维护。

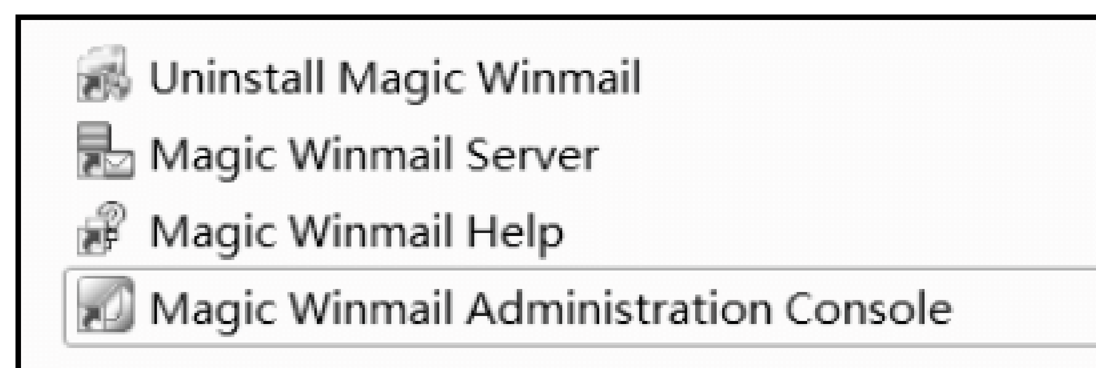


图 2.16　Winmail 的文件列表

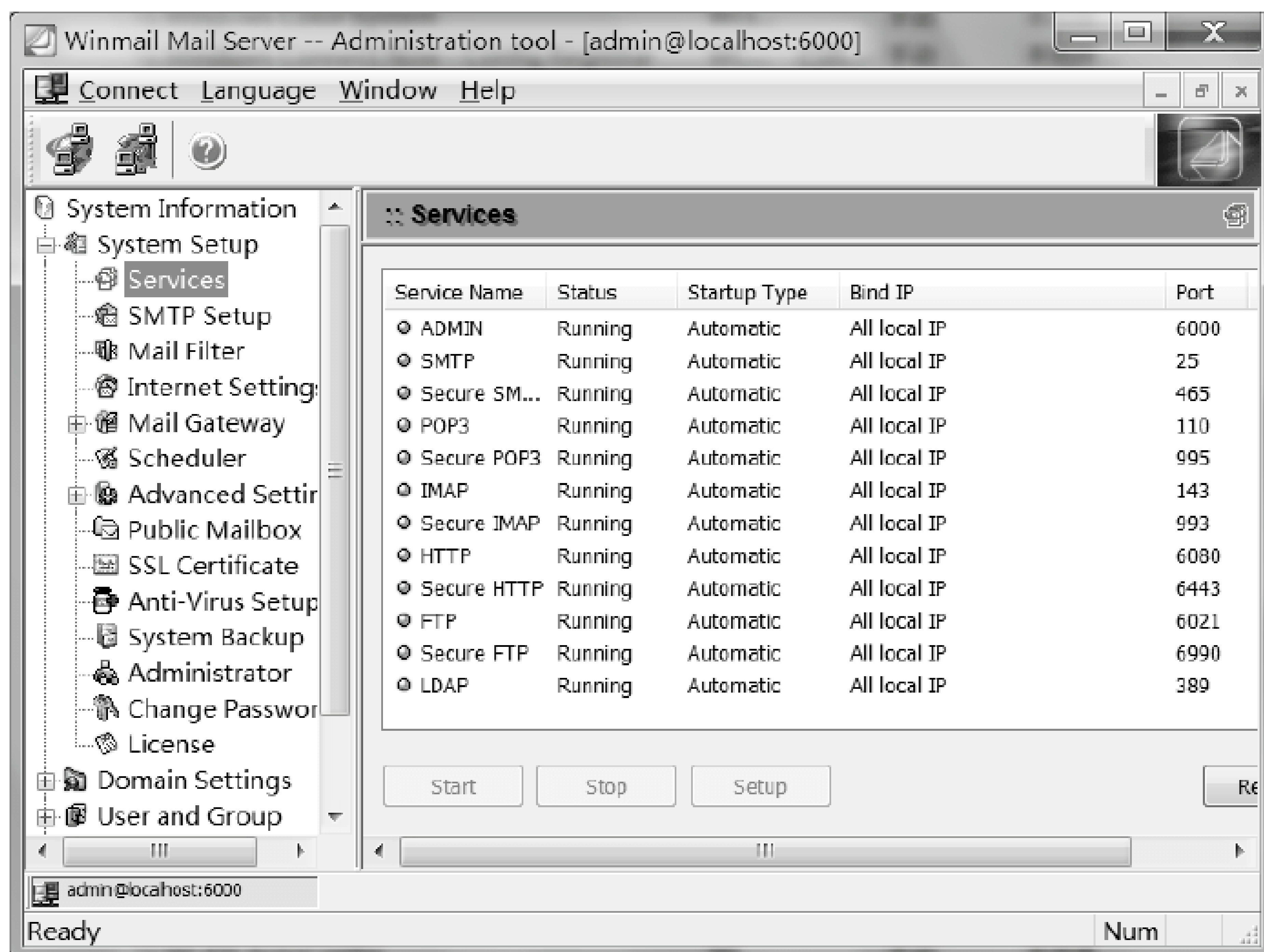


图 2.17　Winmail 邮件服务器的管理器

■ 秘笈心法

心法领悟 040：查看或修改 Winmail 邮件服务器服务的端口。

在 Winmail 邮件服务器的管理器中，选择 System Setup 节点下的 Services 节点，可以看到该邮件服务器的相关服务，如 SMTP、POP3 和 IMAP 协议的服务，在此可以修改相关协议的端口号；可以看到 Services 列表框中包含一个端口号为 6080 的 HTTP 服务，启动这个服务之后，就可以在浏览器中输入 http://localhost:6080/访问 WebMail 了。

2.2　应用 JavaMail 组件发送邮件

JavaMail API 是 Oracle 公司旗下的 Sun 开发团队为方便 Java 开发人员在应用程序中实现邮件发送和接收功

能而提供的一套标准开发包。下面介绍一下如何应用 JavaMail 组件发送邮件。

实例 041　发送普通格式的邮件

光盘位置：光盘\MR\02\041

初级

实用指数：★★★★

实例说明

在日常工作、生活中，经常要发送电子邮件。本实例将介绍如何利用 JSP 开发电子邮件发送程序。运行本实例，在如图 2.18 所示页面中输入邮件相关信息，单击“发送”按钮，当出现“邮件发送成功”的提示信息时，说明电子邮件已成功发送到收件人的邮箱。

图 2.18　发送普通格式的邮件

关键技术

JavaMail 相关类库的文件可以到甲骨文的官方网站（http://www.oracle.com）中下载，目前最新版本的文件名为 javamail-1.4.3.zip。

JavaMail 对 SMTP、POP3、IMAP 提供支持，封装了电子邮件功能中的邮件对象、发送、身份验证、接收等功能。

在发送各种类型的邮件时，主要应用到下面几个类。

- ☑ Session类：要发送邮件，首先需要创建Session类的对象，利用该对象创建邮件对象，指定邮件服务器认证的客户端属性。其类层次结构为javax.mail.Session。
- ☑ InternetAddress类：邮件发送的地址类，其类层次结构为javax.mail.internet.InternetAddress，继承自抽象类javax.mail.Address。
- ☑ MimeMessage类：邮件消息类，其类层次结构为javax.mail.internet.MimeMessage，继承自抽象类javax.mail.Message。
- ☑ Transport类：邮件发送类，其类层次结构为javax.mail.Transport。
- ☑ Authenticator类：授权者类。JavaMail通过使用Authenticator类以用户名、密码的方式访问受到保护的资源，此处“资源”是指邮件服务器。其类层次结构为javax.mail. Authenticator。
- ☑ Store类：用来从邮件服务器上接收邮件，其类层次结构为javax.mail.Store。
- ☑ Folder类：邮件文件夹类，其类层次结构为javax.mail.Folder。

（1）编写一个用于填写邮件发送信息的 JSP 页面 index.jsp，在该页面中添加用于收集邮件发送信息的表单及表单元素。关键代码如下：

```
<form name="form1" method="post" action="mydeal.jsp" onSubmit="return checkform(form1)">
	收件人：<input name="to" type="text" id="to" title="收件人" size="60" >
	发件人：<input name="from" type="text" id="from" title="发件人" size="60">
	密码：<input name="password" type="password" id="password" title="发件人信箱密码" size="60">
	主  题：  <input name="subject" type="text" id="subject" title="邮件主题" size="60">
	内  容：
	<textarea name="content" cols="59" rows="7" class="wenbenkuang" id="content" title="邮件内容"></textarea>
	<input name="Submit" type="submit" class="btn_bg" value="发送">
	<input name="Submit2" type="reset" class="btn_bg" value="重置">
	<input name="Submit3" type="button" class="btn_bg" onClick="window.close();" value="关闭">
</form>
```

（2）创建发送电子邮件的 JSP 页面，完整代码如下：

```
<%@ page contentType="text/html; charset=GBK" language="java"   errorPage="" %>
<%@ page import="java.util.*" %>
<%@ page import ="javax.mail.*" %>
<%@ page import="javax.mail.internet.*" %>
<%@ page import="javax.activation.*" %>
<%
try{
	request.setCharacterEncoding("GBK");
	String from=request.getParameter("from");
	String to=request.getParameter("to");
	String subject=request.getParameter("subject");
	String messageText=request.getParameter("content");
	String password=request.getParameter("password");
	//****如果是在 Internet 上发送电子邮件，使用这段代码将自动生成 SMTP 的主机名称********/
	//int n =from.indexOf('@');
	//int m=from.length() ;
	//String mailserver ="smtp."+from.substring(n+1,m);
	/*************************************************************************/
	String mailserver="localhost";		//在局域网中发送电子邮件，使用这句代码指定 SMTP 服务器
	//建立邮件会话
	Properties pro=new Properties();
	pro.put("mail.smtp.host",mailserver);
	pro.put("mail.smtp.auth","true");
	Session sess=Session.getInstance(pro);
	sess.setDebug(true);
	//新建一个消息对象
	MimeMessage message=new MimeMessage(sess);
	//设置发件人
	InternetAddress from_mail=new InternetAddress(from);
	message.setFrom(from_mail);
	//设置收件人
	InternetAddress to_mail=new InternetAddress(to);
	message.setRecipient(Message.RecipientType.TO ,to_mail);
	//设置主题
	message.setSubject(subject);
	//设置内容
	message.setText(messageText);
	//设置发送时间
	message.setSentDate(new Date());
	//发送邮件
	message.saveChanges();			//保证报头域同会话内容保持一致
	Transport transport =sess.getTransport("smtp");
	transport.connect(mailserver,from,password);
	transport.sendMessage(message,message.getAllRecipients());
	transport.close();
	out.println("<script language='javascript'>alert('邮件已发送！');window.location.href='index.jsp';</script>");
}catch(Exception e){
	System.out.println("发送邮件产生的错误："+e.getMessage());
	}
%>
```

心法领悟 041：设置外网 SMTP 邮箱服务器地址。

如果向外网发送邮件，如向网易 163 邮箱发送邮件，那么需要设置 mail.smtp.host 的值为 163 邮箱服务器的

地址（163 邮箱 SMTP 服务器的地址为 smtp.163.com）。同时，还要设置 mail.smtp.auth 的值为 true，也就是打开 SMTP 协议的认证。

实例 042　发送 HTML 格式的邮件

光盘位置：光盘\MR\02\042

中级

实用指数：★★★

实例说明

HTML 格式的邮件即超文本格式的邮件，比单纯的文本邮件内容更加丰富，更具表现力。运行程序，在如图 2.19 所示页面中输入邮件信息，单击“发送”按钮，即可将邮件发送到收件人的邮箱中。使用 Winmail 打开接收到的邮件，页面效果如图 2.20 所示。



图 2.19　发送 HTML 格式邮件



图 2.20　使用 Winmail 打开 HTML 格式的邮件效果

关键技术

与发送普通格式的邮件相似，不同之处在于定义了 MimeMultipart 对象来存储 HTML 文件的具体内容。在设置内容的同时应该设置对象的格式，本程序中为 text/html; charset=GBK。

MimeMultipart 类的类层次结构是 javax.mail.Internet.MimeMultipart。一般保存电子邮件内容的容器是 Mulitipart 抽象类，它定义了增加、删除及获得电子邮件不同部分内容的方法。由于 Multipart 是抽象类，所以必须使用一个具体的子类，对此 JavaMail API 提供了 javax.mail.Internet.MimeMultipart 类来使用 MimeMessage 对象。

MimeBodyPart 类的类层次结构是 javax.mail.Internet.MimeBodyPart。MimeBodyPart 是 BodyPart 具体用于 MimeMessage 的一个子类。MimeBodyPart 对象代表一个 MimeMessage 对象内容的一部分。每个 MimeBodyPart 被认为由两部分组成：一个 MIME 类型和匹配这个类型的内容。

语法：

```
BodyPart mdp=new MimeBodyPart();              //新建一个存放信件内容的 BodyPart 对象
String messageText="Hello World!";
mdp.setContent(messageText,"text/html");      //定义 MIME 类型为 text/html，并设置 MimeBodyPart 的内容
```

设计过程

（1）编写一个用于填写邮件发送信息的 JSP 页面 index.jsp，在该页面中添加用于收集邮件发送信息的表单及表单元素。关键代码如下：

```
<script src="JS/UBBCode.js"></script>
<form name="form1" method="post" action="mydeal.jsp" onSubmit="return checkform(form1)">
```

```
    收件人地址：
    <input name="to" type="text" id="to" title="收件人" size="94">
    发件人地址：
    <input name="from" type="text" id="from" title="发件人" size="94">
    发件人邮箱密码：
    <input name="password" type="password" id="password" title="发件人邮箱密码" size="94">
    邮件主题：
    <input name="subject" type="text" id="subject" title="邮件主题" size="94">
    <img src="images/UBB/B.gif" width="21" height="20"onClick="bold()"><img src="images/UBB/I.gif" width="21" height="20"
onClick="italic()"><img src="images/UBB/U.gif" width="21" height="20" onClick="underline()"> <img src="images/UBB/left.gif" width="14"
height="14" onClick="left()"> <img src="images/UBB/center.gif" width="14" height="14" onClick="center()"> <img src="images/UBB/right.gif"
width="14" height="14" onClick="right()">
    字体
     <select name="font" class="wenbenkuang" id="font" onChange="txtFont(this.options[this.selectedIndex].value)">
          <option value="宋体" selected>宋体</option>
          <option value="黑体">黑体</option>
          <option value="隶书">隶书</option>
          <option value="华文行楷">行楷</option>
          <option value="楷体_GB2312">行楷</option>
      </select>
      字号
      <select name="size" class="wenbenkuang" onChange="txtFontSize(this.options[this.selectedIndex].value)">
           <option value=1>1</option>
           <option value=2>2</option>
           <option value=3 selected>3</option>
           <option value=4>4</option>
           <option value="5">5</option>
           <option value="6">6</option>
           <option value="7">7</option>
       </select>
       <select onChange="txtColor(this.options[this.selectedIndex].value)" name="color" size="1" class="wenbenkuang" id="color">
            <option selected    value="#000000">默认颜色</option>
            ……              <!--此处省略了其他设置颜色选项的代码-->
            <option style="color:#999999" value="#999999">烟雨蒙蒙</option>
     </select>
邮件内容：
<input type="hidden" name="content" title="邮件内容">
<iframe src="blank.htm" width="560" height="150" class="noborder" id="oEditor"></iframe>
     <input name="Submit" type="submit" class="btn_bg" value="发送">
     <input name="Submit2" type="reset" class="btn_bg" value="重置">
</form>
```

在上面的代码中，应用了自定义的在线文本编辑器。关于在线文本编辑器涉及的 JavaScript 代码被保存到 JS/UBBCode.js 文件中。

（2）编写邮件发送页面 mydeal.jsp，在该页面中实现邮件的发送。具体代码如下：

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ page import="java.util.*"%>
<%@ page import="javax.mail.*" %>
<%@ page import="javax.mail.internet.*"%>
<%@ page import="javax.activation.*" %>
<%
try{
        request.setCharacterEncoding("GBK");
        String from=request.getParameter("from");                    //获取发件人
        String to=request.getParameter("to");                        //获取收件人
        String subject=request.getParameter("subject");              //获取邮件主题
        String messageText=request.getParameter("content");          //获取邮件内容
        String password=request.getParameter("password");            //获取发件人邮箱密码
        //****如果是在 Internet 上发送电子邮件，使用这段代码将自动生成 SMTP 的主机名称********/
        //int n =from.indexOf('@');
        //int m=from.length() ;
        //String mailserver ="smtp."+from.substring(n+1,m);
        /*******************************************************************************/
        String mailserver="localhost"";                              //在局域网中发送电子邮件的，使用这句代码指定 SMTP 服务器
        Properties prop =new Properties();
        prop.put("mail.smtp.host",mailserver);
```

```
            prop.put("mail.smtp.auth","true");
            Session sess =Session.getInstance(prop);
            sess.setDebug(true);
            MimeMessage message=new MimeMessage(sess);
            //给消息对象设置收件人、发件人、主题
            InternetAddress mail_from =new InternetAddress(from);
            message.setFrom(mail_from);
            InternetAddress mail_to =new InternetAddress(to);
            message.setRecipient(Message.RecipientType.TO,mail_to);
            message.setSubject(subject);
            Multipart mul=new MimeMultipart();    //新建一个 MimeMultipart 对象来存放多个 BodyPart 对象
            BodyPart mdp=new MimeBodyPart();    //新建一个存放信件内容的 BodyPart 对象
            mdp.setContent(messageText,"text/html;charset=GBK");
            mul.addBodyPart(mdp);               //将含有信件内容的 BodyPart 加入到 MimeMultipart 对象中
            message.setContent(mul);            //把 mul 作为消息对象的内容
            message.saveChanges();
            Transport transport = sess.getTransport("smtp");
            //以 SMTP 方式登录邮箱，第 1 个参数是发送邮件用的邮件服务器 SMTP 地址，第 2 个参数为用户名，第 3 个参数为密码
            transport.connect(mailserver,from,password);
            transport.sendMessage(message,message.getAllRecipients());
            transport.close();
            out.println("<script language='javascript'>alert('邮件已发送！');window.location.href='index.jsp';</script>");
    }catch(Exception e){
            System.out.println("发送邮件产生的错误："+e.getMessage());
            out.println("<script language='javascript'>alert('邮件发送失败！');window.location.href='index.jsp';</script>");
    }
%>
```

秘笈心法

心法领悟 042：配置 SMTP 服务器端口号。

如果 SMTP 服务器使用的不是默认的端口号，那么还需要通过 mail.smtp.port 指定 SMTP 的端口号。

实例 043　发送带附件的邮件

光盘位置：光盘\MR\02\043

中级

实用指数：★★

实例说明

在实际应用中，有时需要发送带附件的邮件。运行程序，在如图 2.21 所示页面中输入邮件相关信息和附件信息，单击“发送”按钮，当出现“邮件发送成功”的提示信息时，则说明邮件已成功发送到收件人的邮箱中。

图 2.21　带附件的邮件发送程序

一般在设计带有附件的邮件发送程序时可以遵循以下步骤：

（1）发送带有附件的邮件时，需要建立邮件的各个邮件体部分，在第一部分（即邮件内容文字）后增加一个具有 DataHandler 的附件。

（2）如果将文件作为附件发送，则要建立 File DataSource 类型的对象作为附件数据源；如果从 URL 读取数据作为附件发送，则要建立 URLDataSource 类型的对象作为附件数据源。

（3）将数据源（FileDataSource 或是 URLDataSource）对象作为 DataHandler 类构造方法的参数传入，从而建立一个 DataHandler 对象作为数据源的 DataHandler。

（4）将该 DataHandler 设置为邮件体部分的 DataHandler，这样就完成了邮件体与附件之间的关联工作。下

面的工作就是用 BodyPart 的 setFileName()方法设置附件名为原文件名。

（5）将两个邮件体放入 Multipart 中，设置邮件内容为 Multipart，发送邮件。

（1）编写一个用于填写邮件发送信息的 JSP 页面 index.jsp，在该页面中添加用于收集邮件发送信息的表单及表单元素。关键代码如下：

```
<form action="mydeal.jsp" method="post" name="form1" onSubmit="return checkform(form1)">
  收件人：
  <input name="to" type="text" id="to" title="收件人" size="60">
  发件人：
  <input name="from" type="text" id="from" title="发件人" size="60">
  密码：
  <input name="password" type="password" id="password" title="发件人信箱密码" size="60">
  主题：
  <input name="subject" type="text" id="subject" title="邮件主题" size="60">
  附件：
  <input name="adjunct" type="text" id="adjunct" title="附件" size="45" readonly="yes">
  <input name="Submit3" type="button" class="btn_grey" value="上传附件"
              onClick="window.open('upFile.jsp','','width=350,height=150');">
  内容：
  <textarea name="content" cols="59" rows="7" class="wenbenkuang" id="content" title="邮件内容"></textarea>
  <input name="Submit" type="submit" class="btn_bg" value="发送">
  <input name="Submit2" type="reset" class="btn_bg" value="重置">
</form>
```

（2）编写用于收集上传附件信息的表单及表单元素，关键代码如下：

```
<form name="form1" enctype="multipart/form-data" method="post" action="upFile_deal.jsp">
  请选择上传的文件：
  <input name="file" type="file" size="35">
  注：文件大小请控制在 2MB 以内。
  <input name="Submit" type="submit" class="btn_grey" value="提交">
  <input name="Submit2" type="button" class="btn_grey" onClick="window.close()" value="关闭">
</form>
```

（3）编写上传文件的代码，将选择的附件上传到服务器中指定的 upload 文件夹中。此处应用了文件上传组件 jspSmartUpload。具体代码如下：

```
<%@ page contentType="text/html; charset=GBK" language="java"%>
<jsp:useBean id="upFile" scope="page" class="com.jspsmart.upload.SmartUpload" /><%
upFile.initialize(pageContext);
upFile.upload();
System.out.println("文件大小："+upFile.getFiles().getSize());
if(upFile.getFiles().getSize()>2000000){
 System.out.println("<script>alert('您上传的文件太大，不能完成上传！');</script>");
}else{
 String getFileName=upFile.getFiles().getFile(0).getFileName();
 out.println("<script>opener.form1.adjunct.value='"+getFileName+"';window.close();</script>");
 try{
        upFile.save("/upload");
 }catch(Exception e){
        System.out.println("上传文件出现错误："+e.getMessage());
 }
}
%>
```

（4）编写邮件发送页面 mydeal.jsp，在该页面中实现邮件的发送。具体代码如下：

```
<%@ page contentType="text/html; charset=GBK" language="java"%>
<%@ page import="java.util.*"%>
<%@ page import="javax.mail.*"%>
<%@ page import="javax.mail.internet.*"%>
<%@ page import="javax.activation.*"%>
<%
try{
        request.setCharacterEncoding("GBK");
        String from=request.getParameter("from");
        String to=request.getParameter("to");
```

```
		String subject=request.getParameter("subject");
		String content=request.getParameter("content");
		String password=request.getParameter("password");
		String path =request.getParameter("adjunct");
		//****如果是在 Internet 上发送电子邮件，使用这段代码将自动生成 SMTP 的主机名称********/
		//int n =from.indexOf('@');
		//int m=from.length() ;
		//String mailserver ="smtp."+from.substring(n+1,m);
		/**********************************************************************/
		String mailserver="localhost";			//在局域网上发送电子邮件时，使用这句代码指定 SMTP 服务器
		Properties prop =new Properties();
		prop.put("mail.smtp.host",mailserver);
		prop.put("mail.smtp.auth","true");
		Session sess =Session.getInstance(prop);
		sess.setDebug(true);
		MimeMessage message=new MimeMessage(sess);
		//给消息对象设置收件人、发件人、主题
		InternetAddress from_mail =new InternetAddress(from);
		message.setFrom(from_mail);
		InternetAddress to_mail =new InternetAddress(to);
		message.setRecipient(Message.RecipientType.TO,to_mail);
		message.setSubject(subject);
		Multipart mul=new MimeMultipart();		//新建一个 MimeMultipart 对象来存放多个 BodyPart 对象
		BodyPart mdp=new MimeBodyPart();		//新建一个存放信件内容的 BodyPart 对象
		mdp.setContent(content,"text/html;charset=GBK");
		mul.addBodyPart(mdp);				//将含有信件内容的 BodyPart 加入到 MimeMultipart 对象中
		String adjunctname=new Date().getTime()+path.substring(path.lastIndexOf("."));
		//设置信件的附件（用本机上的文件作为附件）
		mdp=new MimeBodyPart();				//新建一个存放附件的 BodyPart
		path=(request.getRealPath("/upload")+"/"+path).replace("\\","/");
		FileDataSource fds=new FileDataSource(path ); //创建附件的数据源对象
		DataHandler handler=new DataHandler(fds);
		mdp.setFileName(adjunctname);
		mdp.setDataHandler(handler);
		mul.addBodyPart(mdp);
		message.setContent(mul);			//把 mul 作为消息对象的内容
		message.saveChanges();
		Transport transport = sess.getTransport("smtp");
		//以 SMTP 方式登录邮箱，第 1 个参数是发送邮件用的邮件服务器 SMTP 地址，第 2 个参数为用户名，第 3 个参数为密码
		transport.connect(mailserver,from,password);
		transport.sendMessage(message,message.getAllRecipients());
		transport.close();
		out.println("<script language='javascript'>alert('邮件已发送！');window.location.href='index.jsp';</script>");
	}catch(Exception e){
		System.out.println("发送邮件产生的错误："+e.getMessage());
		out.println("<script language='javascript'>alert('邮件发送失败！');window.location.href='index.jsp';</script>");
	}
%>
```

秘笈心法

心法领悟 043：构建附件的数据源。

发送附件之前，需要将附件上传到服务器中，然后再创建 javax.Activation.FileDataSource 的对象构建附件的数据源。在创建该数据源对象时，需要在构造方法中指定附件文件的路径，即上传文件所在服务器的路径。

实例 044　群发普通邮件

光盘位置：光盘\MR\02\044

中级　实用指数：★★★

当要给多个收件人发送相同的信息时，如果一个一个地发送，将会大大降低工作效率。此时可以通过邮件群发来解决此问题。运行程序，在如图 2.22 所示页面中输入邮件的相关信息，单击“发送”按钮，当出现“邮

件发送成功”的提示信息时，说明邮件已成功发送到收件人的邮箱中。

图 2.22　群发普通邮件

■ 关键技术

实现邮件群发与实现普通的邮件发送类似，所不同的是在设置收件人时，不再使用 setRecipient()方法，而是使用 setRecipients()方法。下面将对 setRecipients()方法进行介绍。

setRecipients()方法主要用于设置一组收件人的地址，其语法格式如下：

```
message.setRecipients(Message.RecipientType type,Address[] address)
```

参数说明

❶ type：用于设置收件人类型。可以使用以下 3 个常量来区分收件人的类型。

☑ Message.RecipientType.TO：发送。

☑ Message.RecipientType.CC：抄送。

☑ Message.RecipientType.BCC：暗送。

❷ address：用于指定收件人地址。这里为一个 address 数组，可以通过这个数组指定多个收件人地址。

■

（1）编写一个用于填写邮件发送信息的 JSP 页面 index.jsp，在该页面中添加用于收集邮件发送信息的表单及表单元素。关键代码如下：

```
<form name="form1" method="post" action="mydeal.jsp" onSubmit="return checkform(form1)">
收件人：  <input name="to" type="text" id="to" title="收件人" size="60" >
发件人：<input name="from" type="text" id="from" title="发件人" size="60">
密码：<input name="password" type="password" id="password" title="发件人邮箱密码" size="60">
主  题：  <input name="subject" type="text" id="subject" title="邮件主题" size="60">
内  容：
<textarea name="content" cols="59" rows="7" class="wenbenkuang" id="content" title="邮件内容"></textarea>
<input name="Submit" type="submit" class="btn_bg" value="发送">
<input name="Submit2" type="reset" class="btn_bg" value="重置">
<input name="Submit3" type="button" class="btn_bg" onClick="window.close();" value="关闭">
</form>
```

（2）创建发送电子邮件的 JSP 页面 mydeal.jsp。该页面同发送普通邮件类似，只需要将原来的单个收件人修改为现在的多个收件人即可。关键代码如下：

```
//获取多个收件人
String to1=request.getParameter("to1");
String to2=request.getParameter("to2");
String to3=request.getParameter("to3");
//设置收件人
InternetAddress[] to_mail={new InternetAddress(to1),new InternetAddress(to2),new InternetAddress(to3)};
message.setRecipients(Message.RecipientType.TO ,to_mail);
```

秘笈心法

心法领悟 044：邮件的抄送和暗送。

在实际应用中，经常会用到抄送和暗送这两种形式的邮件发送。抄送和暗送虽然都是将一封 E-mail 同时发送到多个信箱，但是二者还是有一定的区别，其区别在于暗送隐藏了其他抄送人的地址，只让收信人看到自己的收信地址，可以起到保密和预防垃圾邮件的作用。

实例 045　群发 HTML 格式的邮件

光盘位置：光盘\MR\02\045

中级

实用指数：★★

实例说明

本实例将介绍如何群发 HTML 格式的邮件。在如图 2.23 所示页面中，输入多个收件人和发件人的邮箱地址，然后输入发件人的邮箱密码、主题和内容，单击“发送”按钮，邮件就会被分别发送到不同收件人的邮箱中。

图 2.23　群发 HTML 格式的邮件

关键技术

群发 HTML 邮件与群发普通邮件类似，需要创建表示多个收件人地址的 InternetAddress 对象数组，再通过 Message 消息对象的 addRecipients()方法添加多个收件人的地址，然后创建 MimeMultipart 对象来存储 HTML 文件的具体内容，最后通过 Transport 对象发送即可。

（1）创建群发 HTML 格式邮件的表单页 index.jsp，并设置表单提交到 Servlet 进行处理，具体代码参见光盘源代码。

（2）创建处理表单信息并实现邮件群发的 Servlet 类 Mail，在 doPost()方法中实现群发 HTML 格式的邮件功能。关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
      response.setContentType("text/html");
      response.setCharacterEncoding("GBK");                          //设置响应的编码格式
      PrintWriter out = response.getWriter();
      try {
            request.setCharacterEncoding("GBK");
            String from = request.getParameter("from");              //获取发件人
            String to1 = request.getParameter("to1");                //获取收件人 1
            String to2 = request.getParameter("to2");                //获取收件人 2
            String to3 = request.getParameter("to3");                //获取收件人 3
            String subject = request.getParameter("subject");        //获取邮件主题
            String messageText = request.getParameter("content");    //获取邮件内容
            String password = request.getParameter("password");      //获取发件人邮箱密码
            String mailserver = "localhost";                         //指定 SMTP 服务器地址
            //建立邮件会话
            Properties props = new Properties();
            props.put("mail.smtp.host", mailserver);                 //指定 SMTP 协议
            props.put("mail.smtp.auth", "true");                     //指定需要向服务器端提交身份认证
```

```
            Session sess = Session.getInstance(props);                          //获取 session
            sess.setDebug(true);                                                //设置调试标志
            MimeMessage message = new MimeMessage(sess);                        //新建一个消息对象
            //设置发件人
            InternetAddress from_mail = new InternetAddress(from);
            message.setFrom(from_mail);
            //设置收件人
            InternetAddress[] to_mail={new InternetAddress(to1),new InternetAddress(to2),new InternetAddress(to3)};
            message.addRecipients(Message.RecipientType.TO ,to_mail);
            message.setSubject(subject);                                        //设置主题
            //设置内容
            Multipart mul = new MimeMultipart();                                //新建一个 MimeMultipart 对象来存放多个 BodyPart 对象
            BodyPart mbp = new MimeBodyPart();                                  //新建一个存放邮件内容的 BodyPart 对象
            mbp.setContent(messageText, "text/html;charset=gbk");
            mul.addBodyPart(mbp);                                               //将包含邮件内容的 BodyPart 添加到 MimeMultipart 对象中
            message.setContent(mul);                                            //设置邮件内容
            message.setSentDate(new Date());                                    //设置发送时间
            //发送邮件
            message.saveChanges();                                              //保证报头域同会话内容保持一致
            Transport transport = sess.getTransport("smtp");
            transport.connect(mailserver, from, password);                      //建立与邮件服务器之间的连接
            transport.sendMessage(message, message.getAllRecipients());         //发送邮件
            transport.close();                                                  //关闭与邮件服务器之间的连接
            out.println("<script language='javascript'>alert('邮件已发送！');"+ "window.location.href='index.jsp';</script>");
        } catch (Exception e) {
            System.out.println("发送邮件产生的错误：" + e.getMessage());
            out.println("<script language='javascript'>alert('邮件发送失败！');"+ "window.location.href='index.jsp';</script>");
        }
}
```

秘笈心法

心法领悟 045：实现邮件群发。

在应用 Message 消息对象设置群发邮件地址时，使用 setRecipients()方法或 addRecipients()方法都可以实现邮件的群发。

实例 046	群发带附件的邮件 光盘位置：光盘\MR\02\046	中级 实用指数：★★

实例说明

本实例将介绍如何群发带附件的邮件。在如图 2.24 所示页面中输入多个收件人和发件人的邮箱地址，然后输入发件人的邮箱密码，选择群发的附件，并输入主题和内容，单击"发送"按钮后，邮件就会被分别发送到不同收件人的邮箱中。

图 2.24　群发带附件的邮件

群发带附件邮件与群发普通邮件类似，只需创建表示多个收件人地址的 InternetAddress 对象数组，然后建立 FileDataSource 类型的对象作为附件数据源，将数据源 FileDataSource 对象作为 DataHandler 类构造方法的参数传入，从而建立一个 DataHandler 对象作为数据源的 DataHandler。将这个 DataHandler 设置为邮件体部分的 DataHandler，这样就完成了邮件体与附件之间的关联工作。最后将两个邮件体放入到 Multipart 中，并设置邮件

内容为 Multipart，即可发送邮件。

设计过程

（1）编写用于群发带附件的邮件的 JSP 页面 index.jsp，关键代码如下：

```
<form name="form1" method="post" action="mydeal.jsp" onSubmit="return checkform(form1)">
收件人 1：  <input name="to1" type="text" id="to" title="收件人" size="60" >
收件人 2：  <input name="to2" type="text" id="to" title="收件人" size="60" >
收件人 3：  <input name="to3" type="text" id="to" title="收件人" size="60" >
发件人：<input name="from" type="text" id="from" title="发件人" size="60">
密码：<input name="password" type="password" id="password" title="发件人邮箱密码" size="60">
主  题：  <input name="subject" type="text" id="subject" title="邮件主题" size="60">
附  件：  <input name="adjunct" type="text" id="adjunct" title="附件" size="45" readonly="yes">
                    <input name="Submit3" type="button" class="btn_grey" value="上传附件"
                        onClick="window.open('upFile.jsp','','width=350,height=150');"></td>
内  容：
<textarea name="content" cols="59" rows="7" class="wenbenkuang" id="content" title="邮件内容"></textarea>
<input name="Submit" type="submit" class="btn_bg" value="发送">
<input name="Submit2" type="reset" class="btn_bg" value="重置">
</form>
```

（2）创建发送电子邮件的 JSP 页面 mydeal.jsp。该页面同发送普通邮件类似，只需将原来的单个收件人修改为现在的多个收件人即可。关键代码如下：

```
//获取多个收件人
String to1=request.getParameter("to1");
String to2=request.getParameter("to2");
String to3=request.getParameter("to3");
//设置收件人
InternetAddress[] to_mail={new InternetAddress(to1),new InternetAddress(to2),new InternetAddress(to3)};
message.setRecipients(Message.RecipientType.TO ,to_mail);
```

秘笈心法

心法领悟 046：群发带多个附件的邮件。

对本实例简单地加以修改，还可以实现群发带多个附件的邮件（应用 jspSmartUpload 组件实现附件上传）。这一部分内容比较简单，读者可自己实现。

实例 047	通过邮箱激活用户的注册 光盘位置：光盘\MR\02\047	中级 实用指数：★★★★

实例说明

为了防止恶意注册，可以通过发送邮件的方法来激活新注册的用户。运行程序，在如图 2.25 所示页面中输入注册信息，然后单击“提交信息”按钮，激活注册用户的邮件就会被发送到用户填写的邮箱中，如图 2.26 所示，通过访问邮件中的地址就可以实现激活用户的操作。

图 2.25　用户注册页面



图 2.26　激活用户注册的邮件内容

关键技术

实现激活用户注册的关键之处是要向用户的邮箱中发送一个 URL 链接地址，该链接包含了用户注册时的用户名和 id。实际上这个链接地址是服务器端的 Servlet 的访问路径，当访问该链接时，就会提交当前包含用户名和 id 的请求参数到 Servlet，然后再通过 Servlet 的相关方法更新当前该用户的激活状态即可。

（1）创建 DBConnection 类，用于获取数据库连接。创建 ActivUserDto 类，用于封装数据信息。

（2）创建 ActivUserDao 类，用于对数据表进行操作。在该类中编写 insert()方法，用于注册新用户，并将新用户的编号返回。关键代码如下：

```
public int insert(ActivUserDto dto) {
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    int id = 0;
    try {
        con = DBConnection.getConnection();
        ps = con.prepareStatement("INSERT INTO tb_user values (default,?,?,?,'0')", PreparedStatement.RETURN_GENERATED_KEYS);
        ps.setString(1, dto.getName());
        ps.setString(2, dto.getPwd());
        ps.setString(3, dto.getMail());
      if (ps.executeUpdate() == 1) {                    //如果插入成功
          rs = ps.getGeneratedKeys();                   //得到自动生成的主键编号
          while (rs.next()) {
              id = rs.getInt(1);
          }
      }
      ……//省略部分代码
```

另外，在该类中编写 activUser()方法，用于按照 id、name 将数据表中的用户账号激活。关键代码如下：

```
public void activUser(Integer id, String name) {
    Connection con = null;
    PreparedStatement ps = null;
    try {
        con = DBConnection.getConnection();
        ps = con.prepareStatement("UPDATE tb_user SET state=1 WHERE id=? and name=?");
        ps.setInt(1, id);
        ps.setString(2, name);
        ps.execute();
        ps.close();
        con.close();
    ……//省略部分代码
```

（3）创建 SendMail 类，在该类中编写 sendMail()方法，用于从资源文件中读取要发送激活邮件所用到的邮箱配置信息，并向用户发送激活邮件。关键代码如下：

```
public void sendMail(String mail,String url) {
      InputStream is = this.getClass().getResourceAsStream("/mailInfo.properties");
      Properties prop = new Properties();
      try {
            prop.load(is);//加载资源文件
      } catch (IOException e1) {
            e1.printStackTrace();
      }
      String msgText = "请单击下面的链接激活用户，如果不能单击请手动复制到地址栏中执行\n" + url;
      String smtpHost = prop.get("smtpHost").toString();          //SMTP 服务器名
      String from = prop.get("mailName").toString();              //发信人地址
      String pwd = prop.get("pwd").toString();                    //密码
      String to = mail;                                           //收信人地址
      Properties props = new Properties();                        //创建 Properties 对象
      props.put("mail.smtp.host", smtpHost);                      //创建邮件服务器
      Session session = Session.getDefaultInstance(props, null);  //取得默认的 Session
      MimeMessage message = new MimeMessage(session);             //创建一条信息，并定义发信人地址和收信人地址
```

```
    try {
        message.setFrom(new InternetAddress(from));
        InternetAddress[] address = {new InternetAddress(to)};
        message.setRecipients(Message.RecipientType.TO, address);
        message.setSubject("激活注册用户");                  //设定主题
        message.setSentDate(new Date());                    //设定发送时间
        message.setText(msgText);                           //把前面定义的 msgText 中的文字设定为邮件正文的内容
        message.saveChanges();                              //保存发送信息
        Transport transport = session.getTransport("smtp"); //协议
        transport.connect(smtpHost, from, pwd);             //发信人地址、用户名、密码
        transport.sendMessage(message, message.getAllRecipients());
        transport.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

（4）创建 RegServlet 类。RegServlet 类用于向数据表中插入新用户信息，同时发送激活邮件的 Servlet。该类首先实现了 doGet()与 doPost()方法，二者分别执行 HTTP 中 get 与 post 类型的请求。在本实例中，这两种类型的请求都通过调用 processRequest()方法来实现业务逻辑（processRequest()方法将用户输入的注册信息插入到数据表中，并为用户发送激活邮件）。关键代码如下：

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    ActivUserDto dto = new ActivUserDto();
    dto.setName(request.getParameter("name"));
    dto.setPwd(request.getParameter("pwd1"));
    dto.setMail(request.getParameter("mail"));
    int id = new ActivUserDao().insert(dto);
    String url = "http://";
    url+=request.getLocalAddr()+":";
    url+=request.getLocalPort();
    url+=request.getContextPath();
    url+="/Activation";
    url += "?id="+id+"&name="+dto.getName();
    new SendMail().sendMail(dto.getMail(), url);
    PrintWriter out = response.getWriter();
    out.print("<h3 align='center'>用户注册完成，激活账号邮件已经发出，请登录您的邮箱按照信中地址激活您的账号</h3>");
    out.print("<br><center><a href='index.jsp'>返回首页</a></center>");
    out.close();
}
```

（5）创建 Activation 类。Activation 类是用来处理用户激活请求的 Servlet。该类首先实现了 doGet()与 doPost()方法，二者分别执行 HTTP 中 get 与 post 类型的请求。在本实例中，这两种类型的请求都通过调用 processRequest()方法来实现业务逻辑（processRequest()方法通过获取 URL 中的参数激活用户）。关键代码如下：

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    Integer id = Integer.valueOf(request.getParameter("id"));
    String name = request.getParameter("name");
    new ActivUserDao().activUser(id, name);
    PrintWriter out = response.getWriter();
    out.print("<br><br>");
    out.print("<center>用户"+name+"已被激活</center>");
    out.print("<br><center><a href='index.jsp'>进入登录页面</a></center>");
    out.close();
}
```

（6）编写 index.jsp 页面，用于登录或进入新用户注册页面，具体代码参见配书光盘。

（7）编写 reg.jsp 页面，用于注册新用户，具体代码参见配书光盘。

（8）在 mailInfo.properties 资源文件中配置用户自己发送激活邮件所用邮箱的主机地址、邮箱账号、邮箱密码，关键代码如下：

```
smtpHost=localhost
mailName=aaa@yahoo.com
pwd=111111
```

秘笈心法

心法领悟 047：处理邮件乱码。

在向用户注册的邮箱中发送激活的 URL 链接时，如果用户注册时输入的用户名为中文，有可能会出现乱码问题，因此在本实例中应用了过滤器来统一字符编码格式。

2.3　应用 JavaMail 组件接收邮件

在实现邮件接收时，经常需要应用 JavaMail API 中的 Session 类、Message 类、Store 类、Folder 类和 Flags 类。下面将通过一些实例详细介绍如何应用 JavaMail 组件接收邮件。

实例 048　应用 POP3 协议接收未读邮件和已读邮件

光盘位置：光盘\MR\02\048

高级

实用指数：★★

实例说明

在开发电子邮件系统时，经常需要实现读取未读邮件和已读邮件的功能。但是 POP3 并不支持查看邮件信息及统计新邮件的功能，因此在使用 JavaMail API 时，如果想获取这些信息，需要自己实现。本实例将介绍如何获取采用 POP3 协议的未读邮件和已读邮件。运行本实例，首先进入邮箱登录页面。在该页面中输入 POP3 服务器名、邮箱名和密码，单击“登录”按钮，即可进入未读邮件列表页面。在该页面中，将分页显示未读邮件列表。单击邮件主题可以查看邮件详细信息，同时将该邮件标记为已读邮件。单击页面左侧的“已读邮件”超链接，将进入到已读邮件页面，显示已读邮件列表，如图 2.27 所示。单击邮件主题，可以查看邮件详细信息。



图 2.27　接收的已读邮件列表

在实现本实例时，最关键的技术就是获取邮件的 UID 和通过 UID 搜索邮件，下面进行详细介绍。

（1）获取邮件的 UID

在实现获取邮件的 UID 时，首先需要获取 POP3Folder 的对象，然后通过 FetchProfile 类优化对指定邮件组成部分的提取，最后通过 POP3Folder 类的 getUID()方法获取邮件 UID。

例如，本实例中获取邮件的 UID 的具体代码如下：

```
final POP3Folder folder = (POP3Folder) store.getFolder("inbox");        //获取 POP3Folder 对象
FetchProfile profile = new FetchProfile();                              //创建 FetchProfile 类的对象
profile.add(UIDFolder.FetchProfileItem.UID);                            //添加获取参数 UID
String uid = folder.getUID(arg0);                                       //获取 UID
```

（2）通过 UID 搜索邮件

JavaMail API 并没有提供通过邮件的 UID 搜索邮件的功能，但提供了 search()方法。通过 Folder 类的 search()方法可以搜索邮件夹中符合条件的邮件。search()方法的原型如下：

```
public Message[] search(SearchTerm term)
```

参数说明

term：用于指定搜索条件使用的 SearchTerm 类。

SearchTerm 类提供了 22 个实现子类用来创建不同的搜索条件，但是并没有提供按 UID 进行搜索的方法。这时可以通过 Java 的匿名内部类来构造一个 SearchTerm 类的对象，用来根据 UID 搜索邮件。

例如，本实例中通过编写匿名内部类实现通过邮件的 UID 搜索邮件的具体代码如下：

```
Message message= folder.search(new SearchTerm() {
 public boolean match(Message arg0) {
        try {
                String uid = folder.getUID(arg0);
                if (messageId.equals(uid))
                        return true;
        } catch (MessagingException e) {
                e.printStackTrace();
        }
        return false;
 }
})[0];
```

（1）编写邮箱登录页面 index.jsp，该页面主要用于收集登录邮箱所需的验证信息。关键代码如下：

```
<form name="form1" method="post" action="ReceiveEmail?action=login" onSubmit="return checkform(form1)">
  POP3 服务器：
  <input name="host" type="text" id="host" size="30"   title="POP3 服务器"> (如：pop3.mrwgh.com)
  邮箱名：
  <input name="username" type="text" id="username" size="30"   title="邮箱名">
  (如：wgh717@mrwgh.com)
  密码：
  <input name="pwd" type="password" id="pwd" size="30" title="邮箱密码">
  <input name="Submit" type="submit" class="btn_bg" value="登录">
  <input name="Submit2" type="reset" class="btn_bg" value="重置">
</form>
```

（2）创建一个接收邮件的 Servlet 实现类 ReceiveEmailServlet，并在该 Servlet 中编写用于登录邮件服务器的方法 login()。在该方法中，首先获取用户输入的邮件服务器名、邮箱名和密码，然后调用 ReceiveEmailDAO 类中的 connectStore()方法建立与邮件服务器的连接，如果连接成功，将获取的主机名、邮箱名和密码保存到 session 中，并将页面重定向到接收邮件的 Servlet 映射地址中，否则将页面重定向到错误提示页。login()方法的具体代码如下：

```
public void login(HttpServletRequest request,HttpServletResponse response){
        String host=request.getParameter("host");                                    //获取邮件服务器名
        String username=request.getParameter("username");                            //获取邮箱名
        String password=request.getParameter("pwd");                                 //获取密码
        Store store=receiveEmailDAO.connectStore(host,username,password);            //建立与邮件接收服务器的连接
        if(store!=null){
                HttpSession session=request.getSession();
                session.setAttribute("host",host);                                   //保存主机名到 session 中
                session.setAttribute("username",username);                           //保存邮箱名到 session 中
                session.setAttribute("pwd",password);                                //保存密码到 session 中
                try {
                        store.close();                                               //关闭与邮件接收服务器的连接
                        request.getRequestDispatcher("ReceiveEmail?action=getEmail&flag=0").forward(request,response);
```

```
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }else{                                                                  //将页面重定向到错误提示页
                request.setAttribute("error", "您输入的服务器、用户名或是密码错误，登录失败！");
                try {
                        request.getRequestDispatcher("error.jsp").forward(request, response);
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

（3）创建 Java 类 ReceiveEmailDAO，在该类中编写用于建立与邮件服务器连接的方法 connectStore()。在该方法中，首先创建采用 POP3 协议接收邮件的 Session；然后通过该 Session 获取 store 对象；再通过调用 store 对象的 connect()方法建立与邮件服务器的连接；最后判断是否已经建立连接，如果是，则关闭与邮件服务器的连接，否则设置 store 为 null，表示连接失败。

（4）在接收邮件的 Servlet 实现类 ReceiveEmailServlet 中，编写用于接收未读邮件和已读邮件的方法 getEmail()。在该方法中，首先获取主机名、邮箱名、密码和标记变量，并将标记变量保存到 HttpServletRequest 对象中；然后获取当前页，并调用 ReceiveEmailDAO 类中的 showEmail()方法根据标记变量 flag 的值获取相应的邮件信息；最后调用保存分页信息的 JavaBean 获取当前页的数据，并将其保存到 HttpServletRequest 对象中。getEmail()方法的具体代码如下：

```
public void getEmail(HttpServletRequest request, HttpServletResponse response) throws IOException {
 HttpSession session = request.getSession();
 String host = session.getAttribute("host").toString();                 //获取主机名
 String username = session.getAttribute("username").toString();         //获取邮箱名
 String password = session.getAttribute("pwd").toString();              //获取密码
 String flag = request.getParameter("flag");                            //获取标记变量
 request.setAttribute("flag", flag);                                    //将标记变量保存到 HttpServletRequest 对象中
 String strPage = (String) request.getParameter("Page");                //获取当前页
 int Page = 1;
 List<ReceiveEmailForm> list = null;
 MyPagination pagination = null;
 if (strPage == null) {
        pagination = new MyPagination();
        list = receiveEmailDAO.showEmail(host, username, password, flag);       //获取邮件信息
        int pagesize =3;                                                //指定每页显示的记录数
        list = pagination.getInitPage(list, Page, pagesize);            //初始化分页信息
        request.getSession().setAttribute("pagination", pagination);
 } else {
        pagination = (MyPagination) request.getSession().getAttribute("pagination");
        Page = pagination.getPage(strPage);
        list = pagination.getAppointPage(Page);                         //获取指定页数据
 }
 request.setAttribute("emailList", list);                               //保存当前页的邮件信息
 request.setAttribute("Page", Page);                                    //保存的当前页码
 //将页面重定向到显示邮件信息的页面
 try {
        request.getRequestDispatcher("emailList.jsp").forward(request,          response);
 } catch (Exception e) {
        e.printStackTrace();
 }
}
```

（5）在 ReceiveEmailDAO 类中添加 showEmail()方法，用于获取邮件信息。在该方法中，首先建立与邮件服务器的连接，并获取邮件夹，以及打开邮件夹；然后调用该类中的 queryIsRead()方法获取已经阅读的邮件 ID 组成的字符串，并将其分割为数组；再根据标记变量 flag 的值搜索未读邮件或已读邮件；最后循环读取符合条件的邮件信息，并保存到 List 集合中。showEmail()方法的具体代码如下：

```
public List<ReceiveEmailForm> showEmail(String host, String username, String password, String flag) {
 List<ReceiveEmailForm> list = new ArrayList<ReceiveEmailForm>();
        Store store = connectStore(host, username, password);           //建立与邮件接收服务器的连接
 try {
```

```
final POP3Folder folder = (POP3Folder) store.getFolder("inbox");
folder.open(Folder.READ_WRITE);                                  //打开邮件夹
FetchProfile profile = new FetchProfile();
//************获取邮件的 UID*************************
profile.add(UIDFolder.FetchProfileItem.UID);
profile.add(FetchProfile.Item.ENVELOPE);
Message[] message = null;
String str_messageId = queryIsRead(username);                    //已经阅读的邮件 ID 组成的字符串
if (!"".equals(str_messageId)) {
        final String[] isRead = str_messageId.split(",");
        java.util.Arrays.sort(isRead);                           //对数组进行排序
        if ("0".equals(flag)) {                                  //未读邮件
                message = folder.search(new SearchTerm() {
                        @Override
                        public boolean match(Message arg0) {
                                try {
                                        String uid = folder.getUID(arg0);
                                        if (java.util.Arrays.binarySearch(isRead,uid)<0)
                                                return true;
                                } catch (MessagingException e) {
                                        e.printStackTrace();
                                }
                                return false;
                        }
                });
        } else {                                                 //已读邮件
                //生成搜索条件
                message = folder.search(new SearchTerm() {
                        @Override
                        public boolean match(Message arg0) {
                                try {
                                        String uid = folder.getUID(arg0);
                                        if (java.util.Arrays.binarySearch(isRead,uid)>=0)
                                                return true;
                                } catch (MessagingException e) {
                                        e.printStackTrace();
                                }
                                return false;
                        }
                });
        }
} else {
        if ("0".equals(flag)) {                                  //未读邮件
                message = folder.getMessages();                  //获取全部邮件
                folder.fetch(message, profile);
        }
}
String mail_content = "";
String mail_attach = "";
Message messageI = null;
/*****************************循环读取邮件信息*****************************/
if(message!=null){
        for (int i = message.length - 1; i >= 0; i--) {
                MimeMessage m = (MimeMessage) message[i];
                ReceiveEmailForm f = new ReceiveEmailForm();
                messageI = message[i];
                f.setMessageId(m.getMessageID());                //邮件 ID

                f.setMessageId(folder.getUID(message[i]));       //邮件 ID
                f.setAddresser(messageI.getFrom()[0].toString()); //发件人
                /*************获取邮件主题*********************************************/
                MimeMessage part=(MimeMessage) message[i];
                String head=part.getHeader("SUBJECT")[0];        //获取邮件的头
                if(head.toLowerCase().startsWith("=?gb")){       //获取 GBK 或 GB2312 编码的邮件主题
                        f.setTitle(messageI.getSubject());
                }else{
                        f.setTitle(new String(messageI.getSubject().getBytes("ISO-8859-1"),"GBK"));      //邮件主题
                }
```

```
                /******************************************************************/
                f.setSendTime(messageI.getSentDate().toLocaleString());              //发送时间
                String[] obj = getMailAttach(part, i);                               //调用获取邮件附件的方法
                message[i].getFlags().getSystemFlags().toString();
                mail_content = obj[0];                                               //邮件内容
                mail_attach = obj[1];                                                //邮件附件
                if (mail_attach == null || mail_attach.equals("")) {
                    mail_attach = "0";
                } else {
                    mail_attach = "1";
                }
                f.setAdjunct(mail_attach);                                           //附件
                list.add(f);                                                         //将邮件信息保存到 List 集合中
            }
        }
        /******************************************************************/
        folder.close(false);                                                         //关闭邮件夹
        store.close();                                                               //关闭与邮件服务器的连接
} catch (Exception e) {
        try {
            e.printStackTrace();
            store.close();                                                           //关闭与邮件服务器的连接
        } catch (MessagingException ex) {
            Logger.getLogger(ReceiveEmailDAO.class.getName()).log(Level.SEVERE, null, ex);
        }
}
return list;
}
```

（6）在 ReceiveEmailDAO 类中添加 queryIsRead()方法，用于从数据库中查询已读邮件的 UID。queryIsRead()方法的具体代码如下：

```
public String queryIsRead(String addressee) {
 String messageId = "";
 String sql = "SELECT * FROM tb_isRead WHERE addressee='" + addressee     + "'";
 ResultSet rs = conn.executeQuery(sql);                                   //执行查询语句
 try {
        while (rs.next()) {
                messageId = messageId + rs.getString(2) + ",";
        }
 } catch (SQLException e) {
        e.printStackTrace();
 }
 conn.close();
 return messageId;
}
```

（7）编写显示邮件列表的页面 emailList.jsp。在该页面中，首先应用<jsp:userBean>动作调用进行分页显示的 JavaBean“MyPagination”，并将 scope 属性设置为 session；然后通过 for 循环语句显示邮件列表（需要注意的是，在显示的过程中，如果有附件属性的值为 1，则还需要显示 标记，标识有附件）；最后，在页面的合适位置调用 MyPagination 的 printCtrl()方法输出分页导航栏。由于 emailList.jsp 页面的代码比较简单，这里不作介绍，具体代码参见配书光盘。

（8）在 ReceiveEmailDAO 类中添加 inIsRead()方法，用于将邮件的 UID 保存到数据库中，标记该邮件已经阅读。inIsRead()方法的具体代码如下：

```
public void inIsRead(String messageId,String addressee){
 //将该邮件的 UID 插入到 tb_isRead 数据表中，表示该邮件已读
 String sql="INSERT INTO tb_isRead (messageId,addressee) VALUES('"+messageId+"','"+addressee+"')";
 conn.executeUpdate(sql);          //执行更新操作
}
```

（9）实现查看邮件详细信息及下载邮件功能。需要注意的是，当用户查看未读邮件的详细信息时，还需要调用 ReceiveEmailDAO 类中的 inIsRead()方法，将该邮件的 UID 保存到数据库，标记该邮件已读。关键代码如下：

```
//当读取未读邮件时，向数据库中插入邮件 ID
if("0".equals(flag)){          //当读取未读邮件时
 receiveEmailDAO.inIsRead(f.getMessageId(),username);
}
```

秘笈心法

心法领悟 048：FetchProfile 类。

FetchProfile 类提供了邮件协议提供者特有的可选参数，用于更有效地实现邮件组成部分的预提取。

实例 049	应用 POP3 协议接收带附件的邮件 光盘位置：光盘\MR\02\049	高级 实用指数：★★

实例说明

在开发电子邮件管理系统时，其中一个很重要的功能就是接收邮件。本实例将介绍如何实现接收带附件的邮件。运行本实例程序，在如图 2.28 所示页面中将分页显示邮件列表；同时，如果哪个邮件带有附件，还将应用 📎 进行标记。在该页面中，单击邮件主题，将进入邮件详细信息页面显示邮件的详细信息，并提供附件下载超链接；单击附件名超链接，即可下载附件。



图 2.28　邮件列表页面

关键技术

本实例主要应用 JavaMail 组件的 Part 接口、Multipart 类和 MimeBodyPart 类实现，下面将分别介绍。

（1）Part 接口

JavaMail API 把邮件正文的各个组成部分及整个邮件都抽象为部件，部件用 javax.mail.Part 接口表示。Part 接口包含 MimePart 和 BodyPart 两个接口，其中 MimePart 接口表示符合 MIME 规范的部件，BodyPart 接口表示可以作为邮件正文组成部分的部件。

（2）Multipart 类

Multipart 类表示复合部件，充当 BodyPart 部件的容器。如果邮件包含多个部件，则应先把这些部件放到一个 Multipart 对象中，然后调用 Message 对象的 setContent(Multipart mp)方法，把这个 Multipart 对象作为邮件的正文。由此可见，通过 Message 对象的 getContent()方法可以获取作为邮件正文的 Multipart 类的对象。

通过 Multipart 类的 getBodyPart()方法可以返回 Multipart 对象中的某个 BodyPart 对象。getBodyPart()方法的原型如下。

```
public BodyPart getBodyPart(int arg0) throws MessagingException
```

参数说明

arg0：用于指定 Multipart 对象的某个位置。

例如，本实例中，获取 Multipart 对象中循环变量 j 指定的 BodyPart 对象的具体代码如下：

```
BodyPart mpart = multipart.getBodyPart(j);
```

通过 Multipart 类的 getCount()方法可以返回 Multipart 对象中保存的 BodyPart 对象的个数。getCount()方法的原型如下：

```
public int getCount() throws MessagingException
```

（3）MimeBodyPart 类

MimeBodyPart 类位于 javax.mail.internet 包中，表示 MIME 邮件中的一个 MIME 消息。MimeBodyPart 类实现 Part 接口，因此具有 Part 接口中定义的方法。

通过 MimeBodyPart 类的 getFileName()方法，可以获取关联的文件名。getFileName()方法的原型如下：

```
public String getFileName() throws MessagingException
```

通过 MimeBodyPart 类的 getDisposition()方法可以获取消息头中的 Content-Disposition 头字段。getDisposition()方法的原型如下：

```
public String getDisposition() throws MessagingException
```

通过 MimeBodyPart 类的 getInputStream()方法可以获取附件输入流。getInputStream()方法的原型如下：

```
public InputStream getInputStream()throws IOException,MessagingException
```

（1）编写邮箱登录页面 index.jsp，该页面主要用于收集登录邮箱所需的验证信息。关键代码如下：

```
<form name="form1" method="post" action="Email?action=login" onSubmit="return checkform(form1)">
  POP3 服务器：
  <input name="host" type="text" class="login" id="host">
  邮箱名：
  <input name="username" type="text" class="login" id="username" >
  密码：
  <input name="pwd" type="password" class="login" id="pwd">
  <input name="Submit" type="submit" class="btn_bg" value="登  录">
  <input name="Submit2" type="reset" class="btn_bg" value="重  置">
</form>
```

（2）创建一个接收邮件的 Servlet 实现类 Email，并在该 Servlet 中编写用于建立与邮件服务器连接的方法 connection()。在该方法中，首先创建采用 POP3 协议接收邮件的 Session；然后通过该 Session 获取 store 对象；再通过调用 store 对象的 connect()方法建立与邮件服务器的连接；最后判断是否已经建立连接，如果是，则关闭与邮件服务器的连接，否则设置 store 为 null，表示连接失败。connection()方法的具体代码如下：

```
public Store connection(String host, String username, String password) {
    String protocol = "pop3";
    Properties prop = new Properties();                              //实例化 Properties 类
    prop.setProperty("mail.store.protocol", "pop3");                 //指定采用 POP3 协议接收邮件
    prop.setProperty("mail.pop3.host", host);                        //指定 POP3 服务器
    Session mailSession = Session.getDefaultInstance(prop, null);    //创建 Session
    mailSession.setDebug(false);                                     //设置调试标志为 false，表示不调试
    Store store = null;
    try {
        store = mailSession.getStore(protocol);                      //获取 store 对象
        store.connect(host, username, password);                     //建立与邮件服务器的连接
    } catch (Exception e) {
        e.printStackTrace();
        if (null != store) {
            if (store.isConnected()) {                               //当 store 已经连接
                try {
                    store.close();                                   //关闭与邮件服务器的连接
                } catch (Exception e1) {
                    e1.printStackTrace();
                }
            }else{
                store = null;
            }
        }
    }
    return store;
}
```

（3）在接收邮件的 Servlet 实现类 Email 中，编写用于登录邮件服务器的方法 login()。在该方法中，首先获取用户输入的邮件服务器名、邮箱名和密码，然后调用步骤（2）中编写的 connection()方法建立与邮件服务器的连接，如果连接成功，将获取的主机名、邮箱名和密码保存到 Session 中，并将页面重定向到接收邮件的 Servlet 映射地址中，否则将页面重定向到错误提示页。login()方法的具体代码如下：

```
public void login(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html;charset=GBK");
    String host = request.getParameter("host");
    String username = request.getParameter("username");
    String password = request.getParameter("pwd");
    Store store = connection(host, username, password);              //建立与邮件服务器的连接
```

```
        if (store.isConnected()) {                              //当 store 已经连接
            try {
                store.close();                                  //关闭与邮件服务器的连接
            } catch (MessagingException ex) {
                Logger.getLogger(Email.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
        if (store != null) {
            HttpSession session = request.getSession();
            session.setAttribute("host", host);                 //保存主机名到 Session 中
            session.setAttribute("username", username);         //保存邮箱名到 Session 中
            session.setAttribute("pwd", password);              //保存密码到 Session 中
            request.getRequestDispatcher("Email?action=receiveEmail").forward(request, response);
        } else {                                                //将页面重定向到错误提示页
            request.setAttribute("error", "您输入的服务器、用户名或是密码错误，登录失败！");
            request.getRequestDispatcher("error.jsp").forward(request, response);
        }
}
```

（4）在接收邮件的 Servlet 实现类 Email 中，编写接收邮件的方法 receiveEmail()。在该方法中，首先获取保存到 Session 中的邮件服务器名、邮箱名、密码和当前页码；然后判断获取的当前页码是否为空，如果为空，则获取全部邮件信息，并保存到 List 集合中，同时初始化分页信息，否则调用保存分页信息的 JavaBean 获取当前页的数据；最后将要显示的邮件信息和当前页码保存到 HttpServletRequest 对象中，并将页面重定向到显示邮件列表页面。receiveEmail()方法的具体代码如下：

```
public void receiveEmail(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html;charset=GBK");
    HttpSession session = request.getSession();
    String host = session.getAttribute("host").toString();              //获取主机名
    String username = session.getAttribute("username").toString();      //获取邮箱名
    String password = session.getAttribute("pwd").toString();           //获取密码
    String strPage = (String) request.getParameter("Page");             //获取当前页
    int Page = 1;
    List list = new ArrayList();
    MyPagination pagination = null;
    if (strPage == null) {
        pagination = new MyPagination();
        /********************获取邮件信息**********************************/
        Store store = null;
        try {
            store = connection(host, username, password);               //建立与邮件服务器的连接
            Folder folder = (Folder) store.getFolder("inbox");
            folder.open(Folder.READ_WRITE);                              //打开邮件夹
            // **************************************************
            Message[] message = folder.getMessages();
            String mail_content = "";
            String mail_attach = "";
            Message messageI = null;
            /*****************************循环读取邮件信息***************************/
            for (int i = message.length - 1; i >= 0; i--) {
                MimeMessage m = (MimeMessage) message[i];
                EmailForm f = new EmailForm();
                messageI = message[i];
                f.setMessageId(m.getMessageID());                        //邮件 ID
                f.setAddresser(messageI.getFrom()[0].toString());        //发件人
                /*************获取邮件主题***************/
                MimeMessage part = (MimeMessage) message[i];
                String head = part.getHeader("SUBJECT")[0];              //获取邮件的头
                if (head.toLowerCase().startsWith("=?gb")) {             //获取 GBK 或 GB2312 编码的邮件主题
                    f.setTitle(messageI.getSubject());
                } else {
                    f.setTitle(new String(messageI.getSubject().getBytes("ISO-8859-1"), "GBK")); //邮件主题
                }
                /****************************************/
                f.setSendTime(messageI.getSentDate().toLocaleString());  //发送时间
                String[] obj = getMailAttach(part, i);                   //调用获取邮件附件的方法
                message[i].getFlags().getSystemFlags().toString();
```

```
                    mail_content = obj[0];                                              //邮件内容
                    mail_attach = obj[1];                                               //邮件附件
                    if (mail_attach == null || mail_attach.equals("")) {
                        mail_attach = "0";
                    } else {
                        mail_attach = "1";
                    }
                    f.setAdjunct(mail_attach);                                          //附件
                    list.add(f);                                                        //将邮件信息保存到 List 集合中
                }
                /*************************************************************************/
                folder.close(false);                                                    //关闭邮件夹
                store.close();                                                          //关闭与邮件服务器的连接
            } catch (Exception e) {
                e.printStackTrace();
                if (store.isConnected()) {                                              //当 store 已经连接
                    try {
                        //当 store 已经连接
                        store.close();                                                  //关闭与邮件服务器的连接
                    } catch (MessagingException ex) {
                        Logger.getLogger(Email.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }
            }
            /*************************************************************************/
            int pagesize =3;                                                            //指定每页显示的记录数
            list = pagination.getInitPage(list, Page, pagesize);                        //初始化分页信息
            request.getSession().setAttribute("pagination", pagination);
        } else {
            pagination = (MyPagination) request.getSession().getAttribute("pagination");
            Page = pagination.getPage(strPage);
            list = pagination.getAppointPage(Page);                                     //获取指定页数据
        }
        request.setAttribute("emailList", list);                                        //保存当前页的邮件信息
        request.setAttribute("Page", Page);                                             //保存的当前页码
        request.getRequestDispatcher("receiveEmail.jsp").forward(request, response);    //将页面重定向到显示邮件信息的页面
    }
```

（5）在接收邮件的 Servlet 实现类 Email 中，编写 getMailAttach()方法，用于获取指定邮件的附件及邮件内容。getMailAttach()方法的具体代码如下：

```
public String[] getMailAttach(Part part, int emailv) throws Exception {
 String contenttype = part.getContentType();                                            //获取内容类型
 int nameindex = contenttype.indexOf("name");                                           //判断内容类型中是否包括 name 关键字
 String fileName1 = "";
 String fileName = "";
 boolean conname = false;
 if (nameindex != -1)
        conname = true;
 if(part.isMimeType("multipart/report")){                                               //获取 UTF-7 编码的邮件内容
        contentMail=decodeStream(part.getInputStream(),"UTF-7");                        //进行转码
 }else{
        if ((part.isMimeType("text/plain") && !conname)        ||
                    (part.isMimeType("text/html") && !conname)) {
                contentMail =part.getContent().toString();
        }else if (part.isMimeType("multipart/*") ) {
                Multipart multipart = (Multipart) part.getContent();
                //获取附件名称（可能包含多个附件）
                for (int j = 0; j < multipart.getCount(); j++) {
                        MimeBodyPart mpart = (MimeBodyPart)multipart.getBodyPart(j);
                        String disposition = mpart.getDisposition();
                        if ((disposition != null)
                                        && ((disposition.equals(Part.ATTACHMENT)) || (disposition
                                                        .equals(Part.INLINE)))) {
                                fileName = mpart.getFileName();                         //获取文件名
                                if (fileName.toLowerCase().indexOf("gbk") != -1) {
                                        fileName = MimeUtility.decodeText(fileName);
                                }else{
                                        fileName=new String(fileName.getBytes("ISO-8859-1"),"GBK");   //对文件名进行转码
```

```
                    }
                    fileName1 = fileName1 + "<a href='DealAttach?" + "bodynum="
                            + j + "&filename=" + fileName + "&emailv=" + emailv
                            + "'>" + fileName + "</a> ";
                }
            }
            int counts = multipart.getCount();
            //通过循环语句获取全部附件
            for (int i = 0; i < counts; i++) {
                getMailAttach(multipart.getBodyPart(i), emailv);
            }
        }
    }
    //将主题内容及附件名称赋值给 String 数组
    String[] obj = {contentMail, fileName1 };
    return obj;
}
```

在编写上面的代码前，需要在 Email 类中定义一个全局变量 contentMail；否则，如果在上面的方法中定义，当邮件包括附件时，将不能正确获取邮件内容。

（6）在接收邮件的 Servlet 实现类 Email 中，编写一个方法 decodeStream()，用于将输入的内容转换为 Java 支持的 UTF-8 编码。该方法的第 1 个参数为 InputStream 对象，用于指定要进行转码的 InputStream 对象；第 2 个参数用于指定要进行转码内容的原编码，这里为 UTF-7。decodeStream()方法的具体代码如下：

```
private String decodeStream(InputStream in, String charset) throws IOException {
    //判断输入的编码格式是否为 UTF-7
    if ("UTF-7".equalsIgnoreCase(charset) || "unicode-1-1-utf-7".equalsIgnoreCase(charset)) {
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        int c;
        while ((c = in.read()) != -1) {
            out.write(c);
        }
        byte[] bytes = out.toByteArray();          //转换为字节数组
        try {
            ByteToCharUTF8 btc = new ByteToCharUTF8();
            char[] chars = new char[bytes.length / 2 + 1];
            btc.convert(bytes, 0, bytes.length, chars, 0, chars.length);
            return new String(chars);
        } catch (Exception e) {
            System.out.println("转码时出现的错误："+e.getMessage());
            //可能会抛出 sun.io.ConversionBufferFullException，那么直接用默认编码解码
            return new String(bytes, "ISO8859-1");
        }
    }
    return "";
}
```

（7）编写显示邮件列表的页面 receiveEmail.jsp。在该页面中，首先应用<jsp:userBean>动作调用进行分页显示的 JavaBean“MyPagination”，并将 scope 属性设置为 session；然后通过 for 循环语句显示邮件列表（需要注意的是，在显示的过程中，如果有附件属性的值为 1，还需要显示 标记，标识有附件）；最后，在页面的合适位置调用 MyPagination 的 printCtrl()方法输出分页导航栏。由于 receiveEmail.jsp 页面的代码比较简单，这里不作介绍，具体代码参见配书光盘。

（8）在接收邮件的 Servlet 实现类 Email 中，编写一个用于获取邮件详细信息的方法 emailDetail()。在该方法中，首先获取邮件服务器名、邮箱名、密码和邮件的 MessageID；然后调用 connection()方法建立与邮件接收服务器的连接，并打开相应的邮件夹；再通过邮件的 MessageID 搜索邮件，并保存到 EmailForm 对象中；最后将获取的邮件信息保存到 HttpServletRequest 对象中，并将页面重定向到显示邮件详细信息的页面。emailDetail()方法的具体代码如下：

```
public void emailDetail(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html;charset=GBK");
    HttpSession session = request.getSession();
    String host = session.getAttribute("host").toString();                    //获取主机名
    String username = session.getAttribute("username").toString();            //获取邮箱名
    String password = session.getAttribute("pwd").toString();                 //获取密码
```

```
        String messageId = request.getParameter("messageId");                    //获取邮件的 MessageID
        EmailForm f = new EmailForm();
        Store store = connection(host, username, password);                     //建立与邮件接收服务器的连接
        try {
            String mail_content = "";
            String mail_attach = "";
            Folder folder = (Folder) store.getFolder("inbox");
            folder.open(Folder.READ_WRITE);
            //通过邮件的 MessageID 搜索邮件
            MessageIDTerm st = new MessageIDTerm(messageId);
            Message[] message = folder.search(st);
            session.setAttribute("message", message);                           //将邮件信息保存到 Session 中
            Message messageI = null;
            if (message.length > 0) {
                MimeMessage m = (MimeMessage) message[0];
                messageI = message[0];
                f.setMessageId(m.getMessageID());                               //邮件 ID
                f.setAddresser(messageI.getFrom()[0].toString());               //发件人
                InternetAddress[] iadd=(InternetAddress[])m.getRecipients(Message.RecipientType.CC);
                if(null!=iadd){
                    String cc=iadd[0].getAddress();
                    f.setCc(cc);//抄送人
                }else{
                    f.setCc("无");
                }
                /*************获取邮件主题*********************************************/
                String head = m.getHeader("SUBJECT")[0];                        //获取邮件的编码格式
                if (head.toLowerCase().startsWith("=?gb")) {                    //获取 GBK 或 GB2312 编码的邮件主题
                    f.setTitle(messageI.getSubject());
                } else {
                    f.setTitle(new String(messageI.getSubject().getBytes("ISO-8859-1"), "GBK")); //邮件主题
                }
                /****************************************************************************/
                f.setSendTime(messageI.getSentDate().toLocaleString());         //发送时间
                String[] obj = getMailAttach((MimeMessage) message[0], 0);      //调用获取邮件附件的方法
                mail_content = obj[0];                                          //邮件内容
                mail_attach = obj[1];                                           //邮件的附件
                if (mail_attach == null || mail_attach.equals("")) {
                    mail_attach = "无";
                }
                f.setContent(mail_content);                                     //设置邮件的详细内容
                f.setAdjunct(mail_attach);                                      //设置附件
            }
            folder.close(false);                                                //关闭邮件夹
            store.close();                                                      //关闭与邮件服务器的连接
        } catch (Exception e) {
            e.printStackTrace();
            try {
                store.close();                                                  //关闭与邮件服务器的连接
            } catch (MessagingException ex) {
                Logger.getLogger(Email.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
        request.setAttribute("emailDetail", f);                                 //保存当前的邮件信息
        request.getRequestDispatcher("email_detail.jsp").forward(request, response);   //将页面重定向到显示邮件详细信息的页面
}
```

（9）编写显示邮件详细信息的页面 email_detail.jsp，在该页面中显示邮件详细信息。由于这部分内容比较简单，这里不作详细介绍，具体代码参见配书光盘。

（10）实现下载附件功能。在查看邮件详细信息时，已经设置了下载附件的超链接。具体代码如下：

```
fileName1 = fileName1 + "<a href='DealAttach?" + "bodynum="+ j + "&filename=" + fileName +
        "&emailv=" + emailv+ "'>" + fileName + "</a> "
```

从上面的代码中可以看出，该超链接地址为一个 Servlet 请求地址。下面就来编写用于下载附件的 Servlet 实现类。在该 Servlet 实现类中，用于处理下载请求的代码全部设置在 doGet()方法中。doGet()方法的具体代码如下：

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
 response.setContentType("text/html;charset=GBK");
       request.setCharacterEncoding("GBK");
 HttpSession session=request.getSession();
 ServletOutputStream out=response.getOutputStream();
 int bodynum=Integer.parseInt(request.getParameter("bodynum"));
 String filename=request.getParameter("filename");
 Message message1[]=(Message[])session.getAttribute("message");
 int i=Integer.parseInt(request.getParameter("emailv"));
 Message message=(Message)message1[i];
        filename=new String(filename.getBytes(),"ISO-8859-1");       //注意此处一定要进行转码，否则下载中文文件名时将出现乱码
 try{
         response.setHeader("Content-Disposition","attachment;filename="+filename);
         Multipart multi=(Multipart)message.getContent();
         MimeBodyPart bodyPart=(MimeBodyPart)multi.getBodyPart(bodynum);
         InputStream is=bodyPart.getInputStream();                  //获取附件的输入流
         int c=0;
         while((c=is.read())!=-1){
                 out.write(c);
         }
 }catch(Exception e){
         System.out.println("doGet 方法产生的错误信息："+e.getMessage());
 }
}
```

秘笈心法

心法领悟 049：List<E>的泛型写法。

在应用 List 集合时，应尽量采用 List<E>的泛型写法，这种写法可以直接定义 List 集合中的对象类型，方便元素的添加和搜索，简化程序代码。

实例 050　应用 IMAP 协议接收未读邮件和已读邮件

光盘位置：光盘\MR\02\050

中级　实用指数：★★

相对于应用 POP3 协议接收邮件而言，使用 IMAP 协议接收未读邮件和已读邮件比较简单。运行本实例，登录邮箱之后，在如图 2.29 所示页面中单击“已读邮件”和“未读邮件”超链接时，即会显示出已读邮件和未读邮件的列表，单击某一邮件的“主题”列超链接，将显示该邮件的详细信息。



图 2.29　应用 IMAP 协议接收未读邮件和已读邮件

关键技术

采用 IMAP 协议接收邮件时，需要经过以下几个关键步骤。

（1）首先需要在 Properties 对象中设置以下键值。

```
Properties prop = new Properties();                     //实例化 Properties 类
prop.setProperty("mail.store.protocol", "imap");        //指定采用 IMAP 协议接收邮件
prop.setProperty("mail.imap.host", host);               //指定 IMAP 服务器
prop.setProperty("mail.imap.class", "com.sun.mail.imap.IMAPStore");
```

（2）在应用邮件会话的 Session 对象的 getStore()方法获取 Store 对象时，getStore()方法的参数为 imap。

（3）利用 Message 消息对象的 isSet()方法区分已读邮件和未读邮件类型。isSet()方法包含一个参数，只要设置参数值为 javax.mail.Flags.Flag.SEEN 即可；如果 isSet()方法返回 true，说明被判断的邮件为已读邮件，否则为未读邮件。

（1）创建 MessageInfo 类，用于重新封装 Message 消息对象返回的邮件信息。关键代码如下：

```
public class MessageInfo {
 private String subject="";                                              //邮件主题
 private String from="";                                                 //邮件发送者地址
 private String to="";                                                   //邮件接收者地址列表
 private String cc="";                                                   //邮件抄送地址列表
 private String bcc="";                                                  //邮件广播地址列表
 private String date;                                                    //邮件发送或接收日期
 private int size;                                                       //邮件大小
 private String text="";                                                 //邮件正文
 private boolean readFlag;                                               //邮件已读标记
 private long uid;                                                       //邮件的 uid
 public MessageInfo(Message msg){                                        //构造方法，对属性进行初始化
      if(msg!=null){
           SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm ");  //日期格式器
           try {
                date = dateFormat.format(msg.getSentDate()!=null?msg.getSentDate():msg.getReceivedDate()); //邮件日期
                subject = msg.getSubject();                              //邮件主题
                size = msg.getSize();                                    //邮件大小
                Object content = msg.getContent();                       //获取邮件正文对象
                String contentType = msg.getContentType();               //获取正文类型
                if(contentType.startsWith("text/plain")){                //普通邮件
                     text=content.toString();                            //邮件正文内容
                }else if(contentType.startsWith("multipart")){           //解析 Multipart 类型的邮件
                     Multipart multipart = (Multipart)content;
                     BodyPart bodyPart = multipart.getBodyPart(0);
                     Scanner in = new Scanner(bodyPart.getInputStream());
                     StringBuffer sb = new StringBuffer();
                     while(in.hasNextLine()){                            //读取邮件正文内容
                          sb.append(in.nextLine());                      //邮件正文内容
                     }
                     text = sb.toString();
                }
                from = convertAddress(msg.getFrom());                    //发送者地址
                to = convertAddress(msg.getRecipients(Message.RecipientType.TO));     //邮件接收者地址
                cc = convertAddress(msg.getRecipients(Message.RecipientType.CC));     //邮件抄送者地址
                bcc = convertAddress(msg.getRecipients(Message.RecipientType.BCC));   //邮件广播地址列表
           } catch (Exception e) {
                e.printStackTrace();
           }
      }
 }
private String convertAddress(Address[] addr){//此方法用于将邮件地址数组转换为使用逗号分隔的字符串
      if(addr==null)
           return "";
      String addressStr="";
```

```
        boolean tf = true;
        for(int i=0;i<addr.length;i++){
                addressStr = addressStr+((tf)? " " : ",")+((InternetAddress)addr[i]).getAddress();
                tf = false;
        }
        return addressStr;
}
……//省略了属性的 getter()和 setter()方法
}
```

（2）创建 EmailServlet 类，编写登录 IMAP 邮件服务器的方法 connectStore()。关键代码如下：

```
public Store connectStore(String host, String username, String password) {
        Properties prop = new Properties();                              //实例化 Properties 类
        prop.setProperty("mail.store.protocol", "imap");                 //指定采用 IMAP 协议接收邮件
        prop.setProperty("mail.imap.host", host);                        //指定 IMAP 服务器
        prop.setProperty("mail.imap.class", "com.sun.mail.imap.IMAPStore");
        prop.setProperty("mail.transport.protocol", "smtp");
        Session mailSession = Session.getDefaultInstance(prop, null);    //创建 Session
        mailSession.setDebug(false);                                     //设置调试标志为 false，表示不调试
        Store store = null;
        try {
                store = mailSession.getStore("imap");                    //获取 store 对象
                store.connect(host, username, password);                 //建立与邮件服务器的连接
        } catch (Exception e) {
                ……//此处省略了非关键代码
        }
        return store;
}
```

（3）在 EmailServlet 类中，编写获取邮件列表的方法，并区分已读邮件和未读邮件，将邮件列表保存在 List 集合中；然后将邮件列表集合保存在 request 范围内；最后将页面请求转发到邮件列表页面。关键代码如下：

```
public void getEmail(HttpServletRequest req,HttpServletResponse res)throws ServletException, IOException{
        HttpSession session = req.getSession();
        String host = session.getAttribute("host").toString();                 //获取主机名
        String userName = session.getAttribute("userName").toString();         //获取邮箱名
        String password = session.getAttribute("pwd").toString();              //获取密码
        String flag = req.getParameter("flag");
        Store store = connectStore(host,userName,password);                    //调用方法连接邮件服务器
        List<MessageInfo> list = new ArrayList<MessageInfo>();                 //创建 List 集合对象，设置元素类型为 MessageInfo
        try {
                IMAPFolder folder= (IMAPFolder)store.getFolder("inbox");       //获取 inbox 邮件夹
                folder.open(Folder.READ_WRITE);                                //打开邮件夹
                if(flag.equals("0")){                                          //用于获取未读邮件
                        Message[] msgs = folder.getMessages();                 //获取所有邮件消息对象
                        for(int i=0;i<msgs.length;i++){
                                long uid = folder.getUID(msgs[i]);/            //获取邮件的 uid
                                MessageInfo myMsg = new MessageInfo(msgs[i]);  //创建自定义的消息对象，对邮件消息进行封装
                                myMsg.setUid(uid);
                                if(!msgs[i].isSet(Flag.SEEN)){                 //未读邮件
                                        list.add(myMsg);                       //添加到 List 集合
                                }
                        }
                }else{
                        Message[] msgs = folder.getMessages();
                        for(int i=0;i<msgs.length;i++){
                                long uid = folder.getUID(msgs[i]);
                                MessageInfo myMsg = new MessageInfo(msgs[i]);
                                myMsg.setUid(uid);
                                if(msgs[i].isSet(Flag.SEEN)){                  //已读邮件
                                        list.add(myMsg);                       //添加到 List 集合
                                }
                        }
                }
        } catch (MessagingException e) {
                e.printStackTrace();
        }
        req.setAttribute("emailList", list);                                   //保存当前页面的邮件信息
```

```
        req.getRequestDispatcher("emailList.jsp").forward(req,res);                    //转发到邮件列表页面
}
```

（4）在 EmailServlet 类中，编写获取邮件详细信息的方法 getMsg()。关键代码如下：

```
private void getMsg(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException{
        String uid = req.getParameter("uid");
        String subject = new String(req.getParameter("subject").getBytes("ISO-8859-1"),"GBK");
        HttpSession session = req.getSession();
        String host = session.getAttribute("host").toString()                  //获取主机名
        String userName = session.getAttribute("userName").toString();         //获取邮箱名
        String password = session.getAttribute("pwd").toString();              //获取密码
        Store store = this.connectStore(host, userName, password);             //连接邮件服务器
        try {
                IMAPFolder folder = (IMAPFolder)store.getFolder("inbox");      //获取 inbox 邮件夹
                folder.open(Folder.READ_WRITE);                                //以读写方式打开邮件夹
                Message[] msgs = folder.getMessages();                         //获取所有邮件的 Message 对象组成的数组
                for(int i = 0;i<msgs.length;i++){
                        if(folder.getUID(msgs[i])==Long.parseLong(uid)&&msgs[i].getSubject().equals(subject)){//根据邮件的 uid 和主题查找邮件
                                MessageInfo msg = new MessageInfo(msgs[i]);
                                req.setAttribute("message", msg);
                                req.setAttribute("flag", "1");
                        }
                }
        } catch (MessagingException e) {
                e.printStackTrace();
        }
        req.getRequestDispatcher("email_detail.jsp").forward(req,res);         //请求转发到邮件详细信息页面
}
```

（5）创建 emailList.jsp 页面，用于显示邮件列表。该页面比较简单，只是从集合中读取邮件列表信息，具体代码参见配书光盘。

（6）创建 email_detail.jsp 页面，用于显示邮件的详细信息。具体代码参见配书光盘。

■ 秘笈心法

心法领悟 050：获取邮件的正文内容。

在获取邮件的正文内容时，需要根据 Message 对象的 getContentType()方法返回邮件正文的 MIME 类型，根据 MIME 类型判断是普通邮件还是其他邮件。如果是普通邮件，那么只要将 Message 消息对象的 getContent()方法返回的对象转换为字符串，即可获取到邮件的正文内容；如果是带附件的邮件，则需要利用 Multipart 对象的 getBodyPart()方法获取 BodyPart 对象，然后调用 getInputStream()方法读取输入流，所读取的输入流的内容也就是邮件的正文内容。

实例 051 应用 IMAP 协议接收带附件的邮件

光盘位置：光盘\MR\02\051

中级

实用指数：★★★★

■ 实例说明

在开发电子邮件管理系统时，其中一个很重要的功能就是接收邮件。本实例将介绍如何应用 IMAP 协议实现接收带附件的邮件。运行本实例程序，在如图 2.30 所示页面中将显示邮件列表；同时，如果哪个邮件带有附件，还将应用 📎 进行标记。在该页面中，单击邮件主题，将进入邮件详细信息页面显示邮件的详细信息，并提供附件下载超链接；单击附件名超链接，即可下载附件。

收件箱

发件人	主题	附件	发送时间
aaa@yahoo.com	发送普通邮件	无	2010/12/17 10:54
aaa@yahoo.com	HTML格式的邮件	无	2010/12/21 10:24
aaa@yahoo.com	发送附件	📎	2010/12/21 10:24
aaa@yahoo.com	邮件群发	无	2010/12/17 13:39
aaa@yahoo.com	群发带附件的邮件	📎	2010/12/21 10:24
aaa@yahoo.com	激活注册用户	无	2010/12/17 15:36
bbb@yahoo.com	附件	📎	2010/12/21 09:46

图 2.30　应用 IMAP 协议接收带附件的邮件

■ 关键技术

本实例主要应用 JavaMail 组件的 Part 接口、Multipart 类和 MimeBodyPart 类来实现，这些接口和类在前面的实例中已经介绍过了，在此不再赘述。

■

（1）创建 MessageInfo 类，用于对 Message 邮件消息进行重新封装。在 MessageInfo 类的构造方法中，对邮件消息的属性进行初始化，如果邮件包含附件，则解析邮件的附件内容。关键代码如下：

```
public class MessageInfo {
    private String subject="";                                          //邮件主题
    private String from="";                                             //邮件发送者地址
    private String to="";                                               //邮件接收者地址列表
    private String cc="";                                               //邮件抄送地址列表
    private String bcc="";                                              //邮件广播地址列表
    private String date;                                                //邮件发送或接收日期
    private int size;                                                   //邮件大小
    private String text="";                                             //邮件正文
    private boolean readFlag;                                           //邮件已读标记
    private boolean hasAdjunct = false;                                 //是否包含附件
    private String adjunct="";                                          //附件名称
    private long uid;                                                   //邮件的 uid
    public MessageInfo(Message msg,int flag){
        if(msg!=null){
            SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm ");    //创建日期格式器
            try {
                date = dateFormat.format(msg.getSentDate()!=null?msg.getSentDate():msg.getReceivedDate());//邮件发送或接收的时间
                subject = msg.getSubject();                             //邮件主题
                size = msg.getSize();                                   //邮件大写
                Object content = msg.getContent();                      //获取邮件正文对象
                String contentType = msg.getContentType();              //获取正文类型
                if(contentType.startsWith("text/plain")){               //普通邮件
                    text=content.toString();
                }else if(contentType.startsWith("multipart")){          //解析 Multipart 类型的邮件
                    Multipart multipart = (Multipart)content;
                    Scanner in = new Scanner(multipart.getBodyPart(0).getInputStream());
                    StringBuffer sb = new StringBuffer();
                    while(in.hasNextLine()){                            //读取邮件正文内容
                        sb.append(in.nextLine());
                    }
                    text = sb.toString();
                    for(int i=0;i<multipart.getCount();i++){            //查找附件
                        BodyPart bodyPart = multipart.getBodyPart(i);
                        String disposition = bodyPart.getDisposition();
                        if ((disposition != null)&& ((disposition.equals(Part.ATTACHMENT)) || (disposition.equals(Part.INLINE)))) {
                            hasAdjunct = true;//确定为附件
                            String fileName = bodyPart.getFileName();   //获取附件文件名
                            if (fileName.toLowerCase().indexOf("gb2312") != -1) {
                                fileName = MimeUtility.decodeText(fileName);
                            }else{
                                fileName=new String(fileName.getBytes("ISO-8859-1"),"gb2312");  //对文件名进行转码
                            }
                            adjunct = "<a href='EmailServlet?action=downAdjunct&bodyId="
                            + i + "&fileName=" + fileName + "&msgId=" + flag
                            + "'>" + fileName + "</a> ";//以附件名为内容，生成一个超链接，链接 URL 为下载附件的地址
                        }

                    }
                }
                from = convertAddress(msg.getFrom());                   //发送者地址
                to = convertAddress(msg.getRecipients(Message.RecipientType.TO));      //邮件接收者地址
                cc = convertAddress(msg.getRecipients(Message.RecipientType.CC));      //邮件抄送者地址
                bcc = convertAddress(msg.getRecipients(Message.RecipientType.BCC));    //邮件广播地址列表
```

```
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	}
……//省略了 getter 和 setter 方法
}
```

（2）创建处理邮件的 Servlet 类 EmailServlet，在该类中编写获取邮件列表的方法 getEmail()，获取邮件的列表信息；再将每个 Message 类型的邮件对象封装成自定义的 MessageInfo 对象，并将 MessageInfo 对象保存在 List 集合中，然后保存在 request 范围内；最后将请求转发到邮件列表页面。关键代码如下：

```
public void getEmail(HttpServletRequest req,HttpServletResponse res)throws ServletException, IOException{
	HttpSession session = req.getSession();
	String host = session.getAttribute("host").toString();			//获取主机名
	String userName = session.getAttribute("userName").toString();		//获取邮箱名
	String password = session.getAttribute("pwd").toString();		//获取密码
	Store store = connectStore(host,userName,password);			//连接邮件服务器
	List<MessageInfo> list = new ArrayList<MessageInfo>();
	try {
		IMAPFolder folder= (IMAPFolder)store.getFolder("inbox");	//获取收件夹对象
		folder.open(Folder.READ_WRITE);					//打开邮件夹
		Message[] msgs = folder.getMessages();				//获取邮件列表
		session.setAttribute("mailList", msgs);				//邮件列表保存在 Session 中
		for(int i=0;i<msgs.length;i++){
			long uid = folder.getUID(msgs[i]);
			MessageInfo myMsg = new MessageInfo(msgs[i],i);		//重新封装邮件信息
			myMsg.setUid(uid);
			list.add(myMsg);					//添加到 List 集合
		}

	} catch (MessagingException e) {
		e.printStackTrace();
	}
	req.setAttribute("emailList", list);					//保存当前页的邮件信息
	req.getRequestDispatcher("receiveEmail.jsp").forward(req,res);
}
```

（3）在 EmailServlet 类中，编写获取邮件详细信息的方法 getMsg()，根据邮件的 UID 和主题查询邮件的详细信息。关键代码如下：

```
private void getMsg(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException{
	String uid = req.getParameter("uid");
	String subject = new String(req.getParameter("subject").getBytes("ISO-8859-1"),"GBK");
	HttpSession session = req.getSession();
	String host = session.getAttribute("host").toString();			//获取主机名
	String userName = session.getAttribute("userName").toString();		//获取邮箱名
	String password = session.getAttribute("pwd").toString();		//获取密码
	Store store = this.connectStore(host, userName, password);
	try {
		IMAPFolder folder = (IMAPFolder)store.getFolder("inbox");
		folder.open(Folder.READ_WRITE);
		Message[] msgs = folder.getMessages();
		for(int i = 0;i<msgs.length;i++){
			if(folder.getUID(msgs[i])==Long.parseLong(uid)&&msgs[i].getSubject().equals(subject)){//根据主题和 UID 查找邮件
				MessageInfo msg = new MessageInfo(msgs[i],i);
				req.setAttribute("message", msg);
			}
		}
	} catch (MessagingException e) {
		e.printStackTrace();
	}
	req.getRequestDispatcher("email_detail.jsp").forward(req,res);
}
```

（4）在 EmailServlet 类中，编写下载附件的方法 downAdjunct()，从 Session 读取邮件信息列表的数组，然后根据指定条件查找包含附件的 Message 对象，实现附件的下载。关键代码如下：

```
private void downAdjunct(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException{
		response.setContentType("text/html;charset=GBK");
```

```
        request.setCharacterEncoding("GBK");
        HttpSession session=request.getSession();
        ServletOutputStream out=response.getOutputStream();
        int bodyId=Integer.parseInt(request.getParameter("bodyId"));                //获取附件的编号
        String fileName=request.getParameter("fileName");                          //获取文件名
        Message msgs[]=(Message[])session.getAttribute("mailList");                //取出邮件列表数组
        int i=Integer.parseInt(request.getParameter("msgId"));                     //获取邮件的编号
        Message message=(Message)msgs[i];                                          //根据编号查找邮件对象
        try{
            response.setHeader("Content-Disposition","attachment;filename="+fileName);  //设置响应报头类型为文件下载
            Multipart multi=(Multipart)message.getContent();
            MimeBodyPart bodyPart=(MimeBodyPart)multi.getBodyPart(bodyId);       //根据编号找到包含附件的对象
            InputStream is=bodyPart.getInputStream();                            //获取附件的输入流
            int c=0;
            while((c=is.read())!=-1){
                out.write(c);                                                    //写入输出流实现下载
            }
        }catch(Exception e){
            e.printStackTrace();
        }
        out.flush();
        out.close();
}
```

（5）创建显示邮件列表的页面 receiveEmail.jsp，在该页面中从 List 集合中取出邮件信息并输出到页面中。具体代码参见配书光盘。

（6）创建显示邮件详细信息的页面 email_detail.jsp，在该页中输出邮件的详细信息。具体代码参见配书光盘。

秘笈心法

心法领悟 051：实现附件的下载。

本实例为了实现附件下载，在 MessageInfo 类的构造方法中直接将附件的名称作为超链接传递到邮件的详细信息页面中。在这个超链接的 URL 中，包含了几个参数，分别是 action（请求 Servlet 时指定调用的方法）、fileName（文件名）、bodyId（当前邮件中包含的附件标记值）、msgId（当前邮件对象的标记值）。当单击附件名称超链接时，会请求 Servlet 类中的下载方法，然后在下载方法中根据这几个参数值即可从邮件列表中查找到需要下载的附件。

2.4　应用 Apache commons-email 组件发送邮件

commons-email 组件是 Apache 组织开发的用于发送邮件的一套 API 类库。该组件依赖于 JavaMail 组件，并对 JavaMail 组件发送邮件进行了更好的封装，因此在应用 commons-email 组件实现收发邮件的功能时，会更加简便。该组件的文件包可在 Apache 官网（www.apache.org）进行下载。

实例 052	发送普通格式的邮件 光盘位置：光盘\MR\02\052	初级 实用指数：★★★★

实例说明

本实例将介绍如何利用 Apache 的 commons-email 组件发送普通格式的邮件。运行本实例，在如图 2.31 所示页面中输入邮件的相关信息，单击“发送”按钮，当出现“邮件发送成功”的提示信息时，说明电子邮件已成功发送到收件人的邮箱。



图 2.31　发送普通格式的邮件

关键技术

发送普通邮件主要用到 commons-email 组件的 SimpleEmail 类，该类继承自 Email 类，主要用于发送普通的文本邮件。SimpleEmail 类中包含了常用的设置发送邮件信息的多个方法，例如设置发送者、接收者、发送时间、发送主题、消息内容等的方法，这些方法均继承自 Email 类。该类的常用方法及说明如表 2.1 所示。

表 2.1　SimpleEmail 类的方法说明

方　　法	描　　述
setHostName(String hostName)	设置邮箱服务器的主机名
setAuthentication(String username,String password)	设置登录邮箱的用户名和密码
setFrom(String from)	设置发件人邮箱地址
addTo(String to)	设置收件人邮箱地址
setSubject(String subject)	设置发送的主题
setSendDate(Date date)	设置发送邮件的时间
setMsg(String message)	设置发送的消息内容
send()	发送邮件

（1）创建发送邮件的表单页 index.jsp，在该页中添加表单元素，用于发送普通的邮件。具体代码参见配书光盘。

（2）创建处理邮件表单页面 mydeal.jsp，在该页中获取表单中的邮件信息；然后创建 comons-email 组件的 SimpleEmail 对象，实现发送普通邮件。关键代码如下：

```
<%
try{
	request.setCharacterEncoding("GBK");
	String from=request.getParameter("from");
	String to=request.getParameter("to");
	String subject=request.getParameter("subject");
	String messageText=request.getParameter("content");
	String password=request.getParameter("password");
	String mailserver="localhost";		//局域网发送邮件时的 SMTP 服务器
	SimpleEmail email = new SimpleEmail();
	email.setHostName(mailserver);		//设置邮件服务器
	email.setAuthentication(from,password);	//设置邮箱用户名和密码
	email.setFrom(from);			//设置发件人地址
	email.addTo(to);			//设置收件人地址
	email.setSubject(subject);		//设置主题
	email.setSentDate(new Date());		//设置发送时间
```

```
            email.setMsg(messageText);                    //设置发送的消息
            email.send();                                 //发送邮件
            out.println("<script language='javascript'>alert('邮件已发送！');window.location.href='index.jsp';</script>");
    }catch(Exception e){
            e.printStackTrace();
            System.out.println("发送邮件产生的错误："+e.getMessage());
            out.println("<script language='javascript'>alert('邮件发送失败！');window.location.href='index.jsp';</script>");
    }
%>
```

秘笈心法

心法领悟 052：设置 SMTP 邮件服务器的端口号。

如果 SMTP 邮件服务器的端口号不是默认的端口号，可以使用 SimpleEmail 类的 setSmtpPort(int port)方法进行设置。

实例 053　发送带多个附件的邮件

光盘位置：光盘\MR\02\053

中级

实用指数：★★★★

实例说明

本实例将介绍如何利用 Apache 的 commons-email 组件实现发送带附件的邮件。运行本实例程序，在如图 2.32 所示页面中输入邮件的相关信息，然后选择要发送的附件，单击“发送”按钮，当出现“邮件发送成功”的提示信息时，说明电子邮件已成功发送到收件人的邮箱。



图 2.32　发送带多个附件的邮件

发送带附件的邮件主要应用 commons-email 组件的 MultiPartEmail 类来实现。这个类继承自 Email 抽象类，主要用于发送包含附件的邮件。可以使用 MultiPartEmail 类中的 attach()方法设置需要发送的附件。该方法包含以下 5 个重载方法：

（1）attach(EmailAttachment attachment)

参数说明

attachment：参数类型为 commons-email 组件的 EmailAttachment 类的对象，该对象表示一个附件。

（2）attach(URL url, String name, String description)

参数说明

❶ url：代表一个统一资源定位符，是指向互联网“资源”的指针，可以直接将网络资源设置为附件。

❷ name：附件的名称。

❸ description：附件的描述。

（3）attach(URL url, String name, String description, String disposition)

参数说明

❶ url：代表一个统一资源定位符，是指向互联网“资源”的指针，可以直接应用网络资源作为附件。

❷ name：附件的名称。

❸ description：附件的描述。

❹ disposition：设置类型。通常设置该参数值为 Email.ATTACHMENTS 类型，即附件类型。

（4）attach(DataSource ds, String name, String description)

参数说明

❶ ds：构建一个本地资源的数据源。参数类型为 javax.activation.DataSource。通过它可以将本地的资源文件作为附件。

❷ name：附件的名称。

❸ description：附件的描述。

（5）attach(DataSource ds, String name, String description, String disposition)

参数说明

❶ ds：构建一个本地资源的数据源。参数类型为 javax.activation.DataSource。通过它可以将本地的资源文件作为附件。

❷ name：附件的名称。

❸ description：附件的描述。

❹ disposition：设置类型。通常设置该参数值为 Email.ATTACHMENTS 类型，即附件类型。

（1）创建发送邮件的表单页 index.jsp，在该页中添加表单元素，用于发送带多个附件的邮件。具体代码参见配书光盘。

（2）创建处理邮件表单页面 mydeal.jsp，在该页中获取表单中的邮件信息；然后创建 comons-email 组件的 EmailAttachment 对象，表示一个附件对象，并设置附件的相关属性；最后通过 MultiPartEmail 对象的 attach()方法实现发送带附件的邮件。关键代码如下：

```
<%
try{
        request.setCharacterEncoding("GBK");
        String from=request.getParameter("from");
        String to=request.getParameter("to");
        String subject=request.getParameter("subject");
        String content=request.getParameter("content");
        String password=request.getParameter("password");
        String filePath1 =request.getParameter("pathStr1");                    //获取本地附件的路径
        String filePath2 =request.getParameter("pathStr2");                    //获取本地附件的路径
        String file1Name = filePath1.substring(filePath1.lastIndexOf("\\")+1);//截取附件名
        String file2Name = filePath2.substring(filePath2.lastIndexOf("\\")+1);//截取附件名
        String mailserver="localhost";                                         //在局域网上发送电子邮件时使用这句代码指定 SMTP 服务器
        MultiPartEmail email = new MultiPartEmail();                           //创建用于发送附件的对象
        email.setHostName(mailserver);                                         //设置邮件服务器主机名
        email.setAuthentication(from,password);                                //设置邮件服务器的邮箱名和密码
        email.setFrom(from);                                                   //发送者地址
        email.addTo(to);                                                       //接收者地址
        email.setSubject(subject);                                             //邮件主题
        email.setSentDate(new Date());                                         //发送时间
        email.setMsg(content);                                                 //发送的消息内容
        EmailAttachment attachment1 = new EmailAttachment();                   //创建附件对象
        BASE64Encoder enc = new BASE64Encoder();                               //处理附件名称的中文乱码问题
        attachment1.setName("=?GBK?B?"+enc.encode(file1Name.getBytes())+"?=");          //设置附件的名称，处理中文时的乱码问题
        attachment1.setPath(filePath1);                                        //设置附件的路径
        email.attach(attachment1);                                             //添加附件
        EmailAttachment attachment2 = new EmailAttachment()                    //创建附件对象
        attachment2.setName("=?GBK?B?"+enc.encode(file2Name.getBytes())+"?=");          //设置附件的名称，处理中文时的乱码问题
        attachment2.setPath(filePath2);                                        //设置附件的路径
        email.attach(attachment2);                                             //添加附件
        email.send();                                                          //发送邮件
        out.println("<script language='javascript'>alert('邮件已发送！');window.location.href='index.jsp';</script>");
}catch(Exception e){
        e.printStackTrace();
        System.out.println("发送邮件产生的错误："+e.getMessage());
```

```
        out.println("<script language='javascript'>alert('邮件发送失败！ ');window.location.href='index.jsp';</script>");
}
%>
```

秘笈心法

心法领悟 053：处理附件名乱码问题。

应用 commons-email 组件发送带附件的邮件时，会出现一个小问题，即当要发送附件的名称为中文时会出现乱码。因此，在发送附件时，需要对附件的名称进行处理。通过以下代码处理后，就不会发生乱码问题了。

```
BASE64Encoder enc = new BASE64Encoder();                                    //处理附件名称的中文乱码问题
attachment.setName("=?GBK?B?"+enc.encode(path.getBytes())+"?=");            //设置附件的名称，处理中文时的乱码问题
```

实例 054　群发普通邮件

光盘位置：光盘\MR\02\054

初级

实用指数：★★★★

实例说明

本实例将介绍如何利用 Apache 的 commons-email 组件实现群发普通的文本邮件。运行本实例，在如图 2.33 所示页面中输入邮件的相关信息，单击“发送”按钮，当出现“邮件发送成功”的提示信息时，说明电子邮件已成功发送到多个收件人的邮箱。

图 2.33　群发普通邮件

关键技术

应用 commons-email 组件群发普通邮件与发送单个邮件类似，同样使用的是 SimpleEmail 类，不同的是需要调用 SimpleEmail 类的 addTo()方法设置多个接收者的邮件地址。例如，有 3 个收件人 a、b 和 c，那么就需要调用 3 次 addTo()方法来分别设置 a、b 和 c 的邮件地址。

（1）创建发送邮件的表单页 index.jsp，在该页中添加表单元素，用于群发普通的邮件。具体代码参见配书光盘。

（2）创建处理邮件表单页面 mydeal.jsp，在该页中获取表单中的邮件信息，然后利用 SimpleEmail 对象的方法实现群发普通邮件。关键代码如下：

```
<%
try{
        request.setCharacterEncoding("GBK");
        String from = request.getParameter("from");
        String to1 = request.getParameter("to1");
        String to2 = request.getParameter("to2");
        String to3 = request.getParameter("to3");
        String subject = request.getParameter("subject");
        String messageText = request.getParameter("content");
        String password = request.getParameter("password");
        String mailserver = "localhost";             //在局域网上发送电子邮件时使用这句代码指定 SMTP 服务器
        SimpleEmail email = new SimpleEmail();
        email.setHostName(mailserver);               //设置邮件服务器
        email.setAuthentication(from,password);      //设置邮箱用户名和密码
        email.setCharset("GB2312");                  //设置字符编码
        email.setFrom(from);                         //设置发件人地址
        email.addTo(to1);                            //设置收件人地址
        email.addTo(to2);                            //设置收件人地址
```

```
        email.addTo(to3);                    //设置收件人地址
        email.setSubject(subject);           //设置主题
        email.setSentDate(new Date());       //设置发送时间
        email.setMsg(messageText);           //设置发送的消息
        email.send();                        //发送邮件
        out.println("<script language='javascript'>alert('邮件已发送！ ');window.location.href='index.jsp';</script>");
}catch(Exception e){
        e.printStackTrace();
        System.out.println("发送邮件产生的错误："+e.getMessage());
        out.println("<script language='javascript'>alert('邮件发送失败！ ');window.location.href='index.jsp';</script>");
}
%>
```

秘笈心法

心法领悟 054：解决邮件内容乱码问题。

在群发邮件时，邮件内容容易出现乱码问题，所以在程序中需要应用 SimpleEmail 类的 setCharset()方法设置字符编码。

实例 055　群发 HTML 格式的邮件

光盘位置：光盘\MR\02\055

初级

实用指数：★★★★

实例说明

本实例将介绍如何利用 Apache 的 commons-email 组件实现群发 HTML 格式的邮件。运行本实例，在如图 2.34 所示页面中输入邮件的相关信息，并设置邮件内容为 HTML 格式，然后单击“发送”按钮，当出现“邮件发送成功”的提示信息时，说明 HTML 格式的电子邮件已成功发送到多个收件人的邮箱。



图 2.34　群发 HTML 格式的邮件

关键技术

应用 commons-email 组件群发 HTML 格式的邮件时，使用的是 HTMLEmail 类。该类同样继承自 Email 抽象类，与使用 SimpleEmail 类发送普通邮件类似，都需要设置邮件的发送者、接收者、发送时间、发送主题等。

有一点不同的是，在调用 HTMLEmail 类的 setMsg()方法设置邮件内容时，可以包含 HTML 的标签。例如：

```
simpleEmail.setMsg("<font color='red'>HTML 邮件的内容</font>")
```

（1）创建发送邮件的表单页 index.jsp，在该页中添加表单元素，用于群发 HTML 格式的邮件。具体代码参见配书光盘。

（2）创建处理邮件表单页面 mydeal.jsp，在该页中获取表单中的邮件信息，然后使用 HTMLEmail 对象的方法实现群发 HTML 格式的邮件。关键代码如下：

```
<%
try{
        request.setCharacterEncoding("GBK");
        String from=request.getParameter("from");
        String to1 = request.getParameter("to1");
        String to2 = request.getParameter("to2");
        String subject=request.getParameter("subject");
```

```
        String messageText=request.getParameter("content");
        String password=request.getParameter("password");
        String mailserver="192.168.1.102";      //在局域网上发送电子邮件时使用这句代码指定 SMTP 服务器
        HtmlEmail email = new HtmlEmail();
        email.setCharset("GB2312");
        email.setHostName(mailserver);           //设置邮件服务器主机名
        email.setAuthentication(from,password);
        email.setFrom(from);                     //设置发件人地址
        email.addTo(to1);                        //设置收件人地址
        email.addTo(to2);                        //设置收件人地址
        email.setSentDate(new Date());           //发送时间
        email.setSubject(subject);               //设置主题
        email.setMsg(messageText);               //设置邮件内容，邮件内容包含 HTML 标签
        email.send();                            //发送邮件
        out.println("<script language='javascript'>alert('邮件已发送！ ');window.location.href='index.jsp';</script>");
}catch(Exception e){
        System.out.println("发送邮件产生的错误："+e.getMessage());
        out.println("<script language='javascript'>alert('邮件发送失败！ ');window.location.href='index.jsp';</script>");
}
%>
```

秘笈心法

心法领悟 055：实现邮件的抄送与暗送。

在发送邮件时，还可以调用 Email 类的 addCc()方法实现抄送，调用 addBcc()方法实现暗送。

实例 056　群发带附件的邮件

光盘位置：光盘\MR\02\056

初级

实用指数：★★★★

实例说明

本实例将介绍如何利用 Apache 的 commons-email 组件实现群发带附件的邮件。运行本实例，在如图 2.35 所示页面中输入邮件的相关信息，然后选择要发送的附件，单击“发送”按钮，当出现“邮件发送成功”的提示信息时，说明电子邮件已成功发送到收件人的邮箱。

图 2.35　群发带附件的邮件

关键技术

应用 commons-email 组件实现群发带附件的邮件很简单，主要利用发送附件的 MultiPartEmail 类来实现。用户只要通过 MultiPartEmail 类的 addTo()方法设置多个收件人邮箱地址，然后通过 attach()方法设置要发送的附件，最后通过 send()方法发送即可。

（1）创建发送邮件的表单页 index.jsp，在该页中添加表单元素，用于群发带附件的邮件。具体代码参见配书光盘。

（2）创建处理邮件表单页面 mydeal.jsp，在该页中获取表单中的邮件信息；然后创建 commons-email 组件的 EmailAttachment 对象，表示一个附件对象，并设置附件的相关属性；最后通过 MultiPartEmail 对象的 attach()方法实现群发带附件的邮件。关键代码如下：

```
<%
try{
        request.setCharacterEncoding("GBK");
```

```
        String from=request.getParameter("from");
        String to1 = request.getParameter("to1");
        String to2 = request.getParameter("to2");
        String subject=request.getParameter("subject");
        String content=request.getParameter("content");
        String password=request.getParameter("password");
        String filePath = request.getParameter("pathStr1");                    //获取附件的路径
        String fileName = filePath.substring(filePath.lastIndexOf("\\")+1);    //截取附件名
        String mailserver="localhost";      //在局域网上发送电子邮件时使用这句代码指定 SMTP 服务器
        MultiPartEmail email = new MultiPartEmail();                           //创建用于发送附件的对象
        email.setHostName(mailserver);                                         //设置邮件服务器主机名
        email.setAuthentication(from,password);                                //设置邮件服务器的邮箱名和密码
        email.setFrom(from);                                                   //发送者地址
        email.addTo(to1);                                                      //接收者地址
        email.addTo(to2);                                                      //接收者地址
        email.setSubject(subject);
        email.setSentDate(new Date());                                         //发送时间
        email.setMsg(content);                                                 //发送的消息内容
        EmailAttachment attachment = new EmailAttachment();                    //创建附件对象
        BASE64Encoder enc = new BASE64Encoder();                               //处理附件名称的中文乱码问题
        attachment.setName("=?GBK?B?"+enc.encode(fileName.getBytes())+"?=");   //设置附件的名称，处理中文时的乱码问题
        attachment.setPath(filePath);                                          //设置附件的路径
        email.attach(attachment);                                              //添加附件
        email.send();                                                          //发送邮件
        out.println("<script language='javascript'>alert('邮件已发送！');window.location.href='index.jsp';</script>");
}catch(Exception e){
        System.out.println("发送邮件产生的错误："+e.getMessage());
}
%>
```

秘笈心法

心法领悟 056：使用另一种方法设置群发的收件人地址。

在实现群发邮件时，也可以通过 setTo(Collection collection)方法设置多个收件人的地址。其中参数 collection 是 java.util.Collection 集合类型，是所有集合类的父接口，因此可以创建一个 ArrayList 集合对象保存收件人的地址。这里需要注意的是，指定集合中的元素类型必须为 javax.mail.internet.InternetAddress 对象类型，也就是需要创建多个 InternetAddress 对象表示收件人的地址，才能实现邮件的群发。

实例 057 通过邮箱激活用户的注册

光盘位置：光盘\MR\02\057

中级

实用指数：★★★★

实例说明

本实例将介绍如何应用 commons-email 组件实现通过发送邮件的方法激活新注册的用户。运行程序，在如图 2.36 所示页面中输入注册信息，然后单击“提交信息”按钮，激活注册用户的邮件就会被发送到用户填写的邮箱中，通过访问邮件中的地址即可实现激活用户的操作，如图 2.37 所示。

通过邮箱激活注册用户

用户名：	测试用户
密码：	••••••
确认密码：	••••••
电子邮箱：	ccc@yahoo.com

提交信息　重新填写

图 2.36　用户注册页面



图 2.37　激活用户注册的邮件内容

关键技术

相对于应用 JavaMail 组件实现发送邮件而言，使用 commons-email 组件实现发送邮件显得更加简单。在实现激活用户注册时，需要应用 commons-email 组件的 SimpleEmail 类来实现向用户的邮箱中发送一个 URL 链接地址。该链接包含了用户注册时的用户名和 id，当访问这个链接时，就会提交当前包含用户名和 id 的请求参数到 Servlet，然后通过 Servlet 的相关方法更新当前该用户的激活状态即可。

（1）创建 DBConnection 类，用于获取数据库连接。创建 ActivUserDto 类，用于封装数据信息。具体代码参见配书光盘。

（2）创建 ActivUserDao 类，用于对数据表进行操作。在该类中编写 insert()方法，用于注册新用户，并将新用户的编号返回。关键代码如下：

```
public int insert(ActivUserDto dto) {
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    int id = 0;
    try {
        con = DBConnection.getConnection();
        ps = con.prepareStatement("INSERT INTO tb_user values (default,?,?,?,'0')", PreparedStatement.RETURN_GENERATED_KEYS);
        ps.setString(1, dto.getName());
        ps.setString(2, dto.getPwd());
        ps.setString(3, dto.getMail());
      if (ps.executeUpdate() == 1) {                        //如果插入成功
          rs = ps.getGeneratedKeys();                       //得到自动生成的主键编号
          while (rs.next()) {
              id = rs.getInt(1);
          }
      }
      ……//省略部分代码
```

另外，在该类中编写 activUser()方法，用于按照 id、name 将数据表中的用户账号激活。关键代码如下：

```
public void activUser(Integer id, String name) {
    Connection con = null;
    PreparedStatement ps = null;
    try {
        con = DBConnection.getConnection();
        ps = con.prepareStatement("UPDATE tb_user SET state=1 WHERE id=? and name=?");
        ps.setInt(1, id);
        ps.setString(2, name);
        ps.execute();
        ps.close();
        con.close();
    ……//省略部分代码
```

（3）创建 SendMail 类，在该类中编写 sendMail()方法，用于从资源文件中读取要发送激活邮件所用到的邮箱配置信息，并向用户发送激活邮件。关键代码如下：

```
public void sendMail(String toAddr,String url) {
      InputStream is = this.getClass().getResourceAsStream("/mailInfo.properties");
      Properties prop = new Properties();
      try {
              prop.load(is);                                  //加载资源文件
      } catch (IOException e1) {
              e1.printStackTrace();
      }
      try{
         String msgText = "请单击下面的链接激活用户，如果不能单击请手动复制到地址栏中执行\n" + url;
         String smtpHost = prop.get("smtpHost").toString();    //SMTP 服务器名
         String from = prop.get("mailName").toString();        //发信人地址
         String pwd = prop.get("pwd").toString();              //密码
         SimpleEmail email = new SimpleEmail();                //创建发送邮件的对象
         email.setCharset("GB2312");                           //设置字符编码
```

```
            email.setHostName(smtpHost);                        //设置邮件服务器
            email.setAuthentication(from, pwd);                 //设置登录邮箱和密码
            email.addTo(toAddr);                                //设置收件人地址
            email.setSentDate(new Date());                      //设置发送时间
            email.setSubject("激活用户注册");                    //设置主题
            email.setMsg(msgText);                              //设置邮件内容
            email.send();                                       //发送邮件
        }catch(Exception ex){
                ex.printStackTrace();
        }
    }
```

（4）创建 RegServlet 类。RegServlet 类用于向数据表中插入新用户信息，同时发送激活邮件的 Servlet。该类首先实现了 doGet()与 doPost()方法，二者分别执行 HTTP 中 get 与 post 类型的请求。在本实例中，这两种类型的请求都通过调用 processRequest()方法来实现业务逻辑（通过 processRequest()方法将用户输入的注册信息插入到数据表中，并为用户发送激活邮件）。关键代码如下：

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        ActivUserDto dto = new ActivUserDto();
        dto.setName(request.getParameter("name"));
        dto.setPwd(request.getParameter("pwd1"));
        dto.setMail(request.getParameter("mail"));
        int id = new ActivUserDao().insert(dto);
        String url = "http://";
        url+=request.getLocalAddr()+":";
        url+=request.getLocalPort();
        url+=request.getContextPath();
        url+="/Activation";
        url += "?id="+id+"&name="+dto.getName();
        new SendMail().sendMail(dto.getMail(), url);
        PrintWriter out = response.getWriter();
        out.print("<h3 align='center'>用户注册完成，激活账号邮件已经发出，请登录您的邮箱按照信中地址激活您的账号</h3>");
        out.print("<br><center><a href='index.jsp'>返回首页</a></center>");
        out.close();
}
```

（5）创建 Activation 类。Activation 类是用来处理用户激活请求的 Servlet。该类首先实现了 doGet()与 doPost()方法，二者分别执行 HTTP 中 get 与 post 类型的请求。在本实例中，这两种类型的请求都通过调用 processRequest()方法来实现业务逻辑（processRequest()方法通过获取 url 中的参数激活用户）。关键代码如下：

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
        Integer id = Integer.valueOf(request.getParameter("id"));
        String name = request.getParameter("name");
        new ActivUserDao().activUser(id, name);
        PrintWriter out = response.getWriter();
        out.print("<br><br>");
        out.print("<center>用户"+name+"已被激活</center>");
        out.print("<br><center><a href='index.jsp'>进入登录页面</a></center>");
        out.close();
}
```

（6）编写 index.jsp 页面，用于登录或进入新用户注册页面。具体代码参见配书光盘。

（7）编写 reg.jsp 页面，用于注册新用户。具体代码参见配书光盘。

（8）在 mailInfo.properties 资源文件中配置用户自己发送激活邮件所用邮箱的主机地址、邮箱账号、邮箱密码，关键代码如下：

```
smtpHost=localhost
mailName=aaa@yahoo.com
pwd=111111
```

心法领悟 057：设置 SMTP 服务器的主机地址。

由于 SMTP 邮件服务器为本机，所以使用 commons-email 组件的 SimpleEmail 类的 setHostName()方法设置

主机名称为 localhost，同样可以通过 IP 地址找到邮件服务器的主机。例如，以局域网中 IP 地址为 192.168.1.102 的机器作为邮件服务器主机，那么 setHostName()方法的参数值就是 192.168.1.102；如果邮件服务器为外网服务器，例如，网易的 126 邮箱服务器，那么 setHostName()方法的参数值自然就是 smtp.126.com，因为 smtp.126.com 是网易 126 邮箱的 SMTP 服务器地址。

2.5　应用 Spring 的 E-mail 抽象层发送邮件

Spring 是一个轻量级的 Java 开发框架，应用它可以简化企业级应用的开发。其核心技术是 IOC（依赖注入）和 AOP（面向切面编程），不过笔者在此不想对 Spring 的核心技术进行深入讲解，仅就如何应用 Spring 提供的组件实现邮件的发送进行介绍。与 commons-email 组件相同，应用 Spring 的 E-mail 抽象层发送邮件，需要有 JavaMail 组件的支持，而且它简化了直接应用 JavaMail 组件发送邮件的功能。Spring 的 E-mail 组件包含在 org.springframework.mail 包中，可以访问其官方网站（www.springsource.org）下载 Spring 框架所有的类库文件。

实例 058　发送普通文本邮件

光盘位置：光盘\MR\02\058

高级

实用指数：★★★★

■ 实例说明

本实例将介绍如何利用 Spring 框架的 E-mail 抽象层实现发送普通格式的邮件。运行本实例程序，在如图 2.38 所示页面中输入邮件的相关信息，然后单击“发送”按钮，当出现“邮件发送成功”的提示信息时，说明电子邮件已成功发送到收件人的邮箱。



图 2.38　发送普通文本邮件

应用 Spring 的 E-mail 抽象层发送普通文本邮件，主要使用 JavaMailSender 接口和 SimpleMailMessage 类来实现。下面进行详细介绍。

（1）JavaMailSender 接口

该接口用于发送邮件。该接口中包含一个名为 JavaMailSenderImpl 的实现类，其常用方法如下。

- ☑ setHost(String host)：设置邮件服务器主机地址。
- ☑ setUsername(String name)：设置登录邮件的用户名。
- ☑ setPassword(String password)：设置登录邮箱的密码。
- ☑ setPort(int port)：设置登录邮件服务器的端口号。
- ☑ setJavaMailProperties(Properties prop)：设置连接邮件服务器的相关配置信息。
- ☑ send(SimpleMailMessage simpleMsg)：发送普通格式的邮件。

（2）SimpleMailMessage 类

该类用来表示一个普通文本邮件的对象。创建这个邮件消息对象后，设置邮件的相关信息，然后通过 JavaMailSender 接口的 send(SimpleMailMessage simpleMsg)方法发送邮件即可。在 SimpleMailMessage 类中主要包含以下几个方法。

- ☑ setFrom(String from)：设置发件人地址。
- ☑ setTo(String to)：设置收件人地址。
- ☑ setCc(String cc)：设置抄送的邮件地址。

- ☑ setBcc(String bcc)：设置暗送的邮件地址
- ☑ setSubject(String subject)：设置邮件主题。
- ☑ setSentDate(Date date)：设置发送邮件的时间。
- ☑ setText(String msg)：设置邮件正文内容。

（1）创建发送邮件的表单页 index.jsp，在该页中添加表单元素，用于发送普通的邮件。具体代码参见配书光盘。

（2）创建 MessageInfo 类，用于封装邮件信息。关键代码如下：

```
public class MessageInfo {
        private String serverHost="";                          //邮件服务器
        private String password;                               //邮箱密码
        private String from = "";                              //发件人地址
        private String to = "";                                //收件人地址
        private Date sendDate;                                 //发送时间
        private String subject;                                //邮件主题
        private String msg="";                                 //消息正文
        private String bcc = "";                               //暗送邮件地址
        private String cc="";                                  //抄送邮件地址
        ……//省略了 getter 和 setter 方法
}
```

（3）自定义发送邮件的类 EmailUtil，在该类中创建一个用于依赖注入的 JavaMailSender 属性，并编写一个 doSend(MessageInfo msg)方法，实现邮件的发送。关键代码如下：

```
public class EmailUtil {
 private JavaMailSender mailSender;                            //注入 Spring E-mail 抽象层的发送邮件对象
 public JavaMailSender getMailSender() {
        return mailSender;
 }
 public void setMailSender(JavaMailSender mailSender) {
        this.mailSender = mailSender;
 }
 public   void doSend(MessageInfo msg){
        SimpleMailMessage message = new SimpleMailMessage();          //创建邮件对象
        message.setFrom(msg.getFrom());                               //设置发送者地址
        message.setTo(msg.getTo());                                   //设置接收者地址
        message.setSubject(msg.getSubject());                         //设置主题
        message.setSentDate(msg.getSendDate());                       //设置发送时间
        message.setText(msg.getMsg());                                //设置消息内容
        JavaMailSenderImpl sender = (JavaMailSenderImpl)mailSender;   //邮件发送对象
        sender.setHost(msg.getServerHost());                          //设置邮件主机地址
        sender.setUsername(msg.getFrom());                            //设置邮箱用户名
        sender.setPassword(msg.getPassword());                        //设置密码
        sender.send(message);                                         //发送邮件
 }
}
```

（4）创建 Spring 的配置文件 applicationContext.xml，对 JavaMailSender 进行初始化配置，并注入到 EmailUtil 对象中。关键代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
 xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:p="http://www.springframework.org/schema/p"
 xsi:schemaLocation="http://www.springframework.org/schema/beans http:  //www.springframework.org/schema/beans/spring-beans-3.0.xsd">
 <bean id="javaMailSender" class="org.springframework.mail.javamail.JavaMailSenderImpl">
        <property name="host" value="localhost" />                    //邮箱服务器地址
        <property name="username" value="aaa@yahoo.com" />            //邮箱用户名
        <property name="password" value="111111" />                   //邮箱密码
        <property name="javaMailProperties"   >                       //邮箱服务器配置属性
               <props>
```

```
                <prop key="mail.smtp.auth">true</prop>                  //登录邮箱身份认证
            </props>
        </property>
</bean>
<bean id="emailUtil" class="com.lh.util.EmailUtil">                     //自定义的发送邮件的类
        <property name="mailSender">                                    //将 JavaMailSender 对象注入
            <ref local="javaMailSender" />
        </property>
</bean>
</beans>
```

（5）创建处理邮件表单页面 mydeal.jsp，在该页中获取表单中的邮件信息，然后装载 Spring 的配置文件，应用 Spring 的 BeanFactory 获取 EmailUtil 对象，并调用 EmailUtil 对象的 doSend()方法实现发送普通文本邮件。关键代码如下：

```
<%
try{
        request.setCharacterEncoding("GBK");
        String from=request.getParameter("from");
        String to=request.getParameter("to");
        String subject=request.getParameter("subject");
        String messageText=request.getParameter("content");
        String password=request.getParameter("password");
        String mailserver="localhost";                                  //局域网发送邮件时的 SMTP 服务器
        MessageInfo message = new MessageInfo();                        //封装邮件信息的对象
        message.setServerHost(mailserver);
        message.setPassword(password);
        message.setFrom(from);
        message.setTo(to);
        message.setSubject(subject);
        message.setSendDate(new Date());
        message.setMsg(messageText);
        Resource resource = new ClassPathResource("applicationContext.xml");   //装载配置文件
        BeanFactory factory = new XmlBeanFactory(resource);
        EmailUtil emailUtil = (EmailUtil)factory.getBean("emailUtil");  //获取发送邮件的对象
        emailUtil.doSend(message);                                      //发送邮件
        out.println("<script language='javascript'>alert('邮件已发送！');window.location.href='index.jsp';</script>");
}catch(Exception e){
        e.printStackTrace();
        System.out.println("发送邮件产生的错误："+e.getMessage());
        out.println("<script language='javascript'>alert('邮件发送失败！');window.location.href='index.jsp';</script>");
}
%>
```

秘笈心法

心法领悟 058：初始化 JavaMailSender 对象。

在初始化 JavaMailSender 对象的配置时，应用的是 Spring 的配置文件 applicationContext.xml；然后再对 EmailUtil 对象的<Bean>进行配置，并将 JavaMailSender 对象注入到 EmailUtil 对象中。当应用程序加载 Spring 容器后，Spring 的 BeanFactory 工厂会根据 applicationContext.xml 的配置自动创建 JavaMailSender 对象，而并不需要通过 new 关键字创建这些对象。

实例 059	发送 HTML 格式的邮件 光盘位置：光盘\MR\02\059	高级 实用指数：★★★★

实例说明

本实例将介绍如何利用 Spring 的 E-mail 抽象层实现发送 HTML 格式的邮件。运行本实例，在如图 2.39 所示页面中输入邮件的相关信息，并设置邮件内容为 HTML 格式，然后单击“发送”按钮，当出现“邮件发送成功”的提示信息时，说明 HTML 格式的电子邮件已成功发送到收件人的邮箱。



图 2.39　发送 HTML 格式的邮件

关键技术

在 Spring 的 E-mail 抽象层中，提供了一个名为 MimeMessageHelper 的邮件类，通过它可以设定更加丰富的邮件内容，如设定 HTML 格式的邮件内容和带附件的邮件。在应用 MimeMessageHelper 之前，需要通过 JavaMailSender 对象的 createMimeMessage()方法创建一个 javax.mail.internet.MimeMessage 类型的邮件对象，然后使用 MimeMessageHelper 对象封装 MimeMessage 对象，最后通过 JavaMailSender 对象的 send(MimeMessage msg)方法完成发送。

在 MimeMessageHelper 类中，主要包含以下几个常用方法。

（1）public MimeMessageHelper(MimeMessage msg , boolean multipart , String encoding)：类的构造方法。其中参数说明如下。

❶ msg：类型为 javax.mail.internet.MimeMessage 邮件对象。

❷ multipart：邮件正文类型是否为 Multipart 类型。

❸ encoding：设置字符编码格式。

（2）setFrom(String from)：设置发件人地址。

（3）setTo(String to)：设置收件人地址。

（4）setCc(String cc)：设置抄送的邮件地址。

（5）setBcc(String bcc)：设置暗送的邮件地址。

（6）setSubject(String subject)：设置邮件主题。

（7）setSentDate(Date date)：设置发送邮件的时间。

（8）setText(String msg , boolean html)：设置邮件正文内容。当html参数值为true时，表示HTML格式的内容。

（1）创建发送邮件的表单页 index.jsp，在该页中添加表单元素，用于发送 HTML 格式的邮件。具体代码参见配书光盘。

（2）创建 MessageInfo 类，用于封装邮件信息。关键代码如下：

```
public class MessageInfo {
        private String serverHost="";        //邮件服务器
        private String password;             //邮箱密码
        private String from = "";            //发件人地址
        private String to = "";              //收件人地址
        private Date sendDate;               //发送时间
        private String subject;              //邮件主题
        private String msg="";               //消息正文
```

```
        private String bcc = "";                                    //暗送邮件地址
        private String cc="";                                       //抄送邮件地址
        ……//省略了 getter 和 setter 方法
}
```

（3）自定义发送邮件的类 EmailUtil，在该类中创建一个用于依赖注入的 JavaMailSender 属性，并编写一个 doSend(MessageInfo msg)方法，实现 HTML 格式的邮件发送。关键代码如下：

```
public class EmailUtil {
 private JavaMailSender mailSender;                                //注入 Spring E-mail 抽象层的发送邮件对象
 public JavaMailSender getMailSender() {
        return mailSender;
 }
 public void setMailSender(JavaMailSender mailSender) {
        this.mailSender = mailSender;
 }
 public   void doSend(MessageInfo msg){
        MimeMessage mimeMessage = mailSender.createMimeMessage();          //创建消息对象
        MimeMessageHelper messageHelper = null;                            //创建用于发送 HTML 格式邮件的对象
        try{
               messageHelper = new MimeMessageHelper(mimeMessage,true,"UTF-8");
               messageHelper.setFrom(msg.getFrom());                      //设置发件人地址
               messageHelper.setTo(msg.getTo());                          //设置收件人地址
               messageHelper.setSubject(msg.getSubject());                //设置主题
               messageHelper.setSentDate(msg.getSendDate());              //设置发送时间
               messageHelper.setText(msg.getMsg(),true);                  //设置邮件正文，正文内容包含 HTML 格式
        }catch(Exception ex){
               ex.printStackTrace();
        }
        JavaMailSenderImpl sender = (JavaMailSenderImpl)mailSender;        //邮件发送对象
        sender.setHost(msg.getServerHost());                               //设置邮件主机地址
        sender.setUsername(msg.getFrom());                                 //设置邮箱用户名
        sender.setPassword(msg.getPassword());                             //设置密码
        sender.send(mimeMessage);
 }
}
```

（4）创建 Spring 的配置文件 applicationContext.xml，对 JavaMailSender 进行初始化配置，并注入到 EmailUtil 对象中。具体代码参见配书光盘。

（5）创建处理邮件表单页面 mydeal.jsp，在该页中获取表单中的邮件信息，然后装载 Spring 的配置文件，使用 Spring 的 BeanFactory 获取 EmailUtil 对象，并调用 EmailUtil 对象的 doSend()方法实现发送 HTML 格式的邮件。关键代码如下：

```
<%
try{
        request.setCharacterEncoding("GBK");
        String from=request.getParameter("from");
        String to=request.getParameter("to");
        String subject=request.getParameter("subject");
        String messageText=request.getParameter("content");
        String password=request.getParameter("password");
        String mailserver="localhost";                         //在局域网上发送电子邮件时使用这句代码指定 SMTP 服务器
        MessageInfo message = new MessageInfo();               //封装邮件信息的对象
        message.setServerHost(mailserver);
        message.setPassword(password);
        message.setFrom(from);
        message.setTo(to);
        message.setSubject(subject);
        message.setSendDate(new Date());
        message.setMsg(messageText);
        Resource resource = new ClassPathResource("applicationContext.xml");   //装载配置文件
        BeanFactory factory = new XmlBeanFactory(resource);
        EmailUtil emailUtil = (EmailUtil)factory.getBean("emailUtil");         //获取发送邮件的对象
        emailUtil.doSend(message);                                             //发送 HTML 邮件
        out.println("<script language='javascript'>alert('邮件已发送！');window.location.href='index.jsp';</script>");
}catch(Exception e){
        System.out.println("发送邮件产生的错误："+e.getMessage());
```

```
        out.println("<script language='javascript'>alert('邮件发送失败！');window.location.href='index.jsp';</script>");
}
%>
```

秘笈心法

心法领悟 059：发送 HTML 格式的邮件问题。

在应用 MimeMessageHelper 对象设置 HTML 格式的邮件内容时，需要注意的是应该调用 setText(String msg , boolean html)方法，并设置第 2 个布尔类型的参数值为 true，否则在发送邮件时会被认为是普通的文本邮件。

实例 060 发送带附件的邮件

光盘位置：光盘\MR\02\060

高级

实用指数：★★★★

实例说明

本实例将介绍如何利用 Spring 的 E-mail 抽象层实现发送带附件的邮件。运行本实例，在如图 2.40 所示页面中输入邮件的相关信息，然后选择要发送的附件，单击“发送”按钮，当出现“邮件发送成功”的提示信息时，说明电子邮件已成功发送到收件人的邮箱。



图 2.40　发送带附件的邮件

在实例 059 中介绍了应用 MimeMessageHelper 类可以设定更加丰富的邮件内容，其实例功能并不止于此。它不仅可以设定 HTML 格式的邮件内容，而且还可以设置邮件的附件。通常利用 MimeMessageHelper 类的 addAttachment()方法来为邮件添加附件，该方法包含以下几个重载的方法。

（1）messageHelper.addAttachment(String fileName, DataSource dataSource)

参数说明

❶ fileName：设置附件的名称。

❷ dataSource：设置附件的数据源。

（2）messageHelper.addAttachment(String fileName, File file)

参数说明

❶ fileName：设置附件的名称。

❷ file：应用 java.io.File 创建附件对象。

（3）messageHelper.addAttachment(String fileName, InputStreamSource inputStreamSource)

参数说明

❶ fileName：设置附件的名称。

❷ inputStreamSource：通过输入流对象设置附件的数据源。

（4）messageHelper.addAttachment(String fileName, InputStreanSource inputStreamSource,String contentType)

参数说明

❶ fileName：设置附件的名称。

❷ inputStreamSource：通过输入流对象设置附件的数据源。

❸ contentType：设置邮件正文的 MIME 类型。

（1）创建发送邮件的表单页 index.jsp，在该页中添加表单元素，用于发送带附件的邮件。具体代码参见配书光盘。

（2）创建 MessageInfo 类，用于封装邮件信息。关键代码如下：

```
public class MessageInfo {
        private String serverHost="";                              //邮件服务器
        private String password;                                   //邮箱密码
        private String from = "";                                  //发件人地址
        private String to = "";                                    //收件人地址
        private Date sendDate;                                     //发送时间
        private String subject;                                    //邮件主题
        private String msg="";                                     //消息正文
        private String bcc = "";                                   //暗送邮件地址
        private String cc="";                                      //抄送邮件地址
        private String fileName;                                   //附件名称
        private String filePath                                    //附件路径
        ……//省略了 getter 和 setter 方法
}
```

（3）自定义发送邮件的类 EmailUtil，在该类中创建一个用于依赖注入的 JavaMailSender 属性，并编写一个 doSend(MessageInfo msg)方法，实现带附件的邮件发送。关键代码如下：

```
public class EmailUtil {
 private JavaMailSender mailSender;                                //注入 Spring E-mail 抽象层的发送邮件对象
 public JavaMailSender getMailSender() {
        return mailSender;
 }
 public void setMailSender(JavaMailSender mailSender) {
        this.mailSender = mailSender;
 }
 public   void doSend(MessageInfo msg){
        MimeMessage mimeMessage = mailSender.createMimeMessage();  //创建消息对象
        MimeMessageHelper messageHelper = null;                    //创建用于封装邮件的对象
        try{
               messageHelper = new MimeMessageHelper(mimeMessage,true,"UTF-8");
               messageHelper.setFrom(msg.getFrom());               //设置发件人地址
               messageHelper.setTo(msg.getTo());                   //设置收件人地址
               messageHelper.setSubject(msg.getSubject());         //设置主题
               messageHelper.setSentDate(msg.getSendDate());       //设置发送时间
               messageHelper.setText(msg.getMsg(),true);           //设置邮件正文，正文内容包含 HTML 格式
               BASE64Encoder enc = new BASE64Encoder();            //此处用于处理附件名乱码问题
               String fileName = "=?GBK?B?"+enc.encode(msg.getFileName().getBytes())+"?=";
               messageHelper.addAttachment(fileName, new FileDataSource(msg.getFilePath()));//添加附件
        }catch(Exception ex){
               ex.printStackTrace();
        }
        JavaMailSenderImpl sender = (JavaMailSenderImpl)mailSender;  //邮件发送对象
        sender.setHost(msg.getServerHost());                       //设置邮件主机地址
        sender.setUsername(msg.getFrom());                         //设置邮箱用户名
        sender.setPassword(msg.getPassword());                     //设置密码
        sender.send(mimeMessage);
 }
}
```

（4）创建 Spring 的配置文件 applicationContext.xml，对 JavaMailSender 进行初始化配置，并注入到 EmailUtil 对象中。具体代码参见配书光盘。

（5）创建处理邮件表单页面 mydeal.jsp，在该页中获取表单中的邮件信息，然后装载 Spring 的配置文件，应用 Spring 的 BeanFactory 获取 EmailUtil 对象，并调用 EmailUtil 对象的 doSend()方法实现发送 HTML 格式的邮件。关键代码如下：

```
<%
try{
        request.setCharacterEncoding("GBK");
```

```
        String from=request.getParameter("from");
        String to=request.getParameter("to");
        String subject=request.getParameter("subject");
        String messageText=request.getParameter("content");
        String password=request.getParameter("password");
        String filePath =request.getParameter("pathStr");                    //获取本地附件的路径
        String fileName = filePath.substring(filePath.lastIndexOf("\\")+1);   //截取附件名
        String mailserver="localhost";                                        //在局域网上发送电子邮件时使用这句代码指定 SMTP 服务器
        MessageInfo message = new MessageInfo();                              //封装邮件信息的对象
        message.setServerHost(mailserver);
        message.setPassword(password);
        message.setFrom(from);
        message.setTo(to);
        message.setSubject(subject);
        message.setSendDate(new Date());
        message.setMsg(messageText);
        message.setFileName(fileName);
        message.setFilePath(filePath);
        Resource resource = new ClassPathResource("applicationContext.xml");  //装载配置文件
        BeanFactory factory = new XmlBeanFactory(resource);
        EmailUtil emailUtil = (EmailUtil)factory.getBean("emailUtil");        //获取发送邮件的对象
        emailUtil.doSend(message);                                            //发送邮件
        out.println("<script language='javascript'>alert('邮件已发送！');window.location.href='index.jsp';</script>");
}catch(Exception e){
        System.out.println("发送邮件产生的错误："+e.getMessage());
        out.println("<script language='javascript'>alert('邮件发送失败！');window.location.href='index.jsp';</script>");
}
%>
```

秘笈心法

心法领悟 060：MimeMessageHelper 类的 addAttachment()方法。

addAttachment()方法包含了多个重载方法，在实际应用中，可以根据不同的需求选择合适的重载方法添加附件。

实例 061　群发普通文本邮件

光盘位置：光盘\MR\02\061

高级

实用指数：★★★★

实例说明

本实例将介绍如何利用 Spring 的 E-mail 抽象层实现群发普通的文本邮件。运行本实例，在如图 2.41 所示页面中输入邮件的相关信息，单击“发送”按钮，当出现“邮件发送成功”的提示信息时，说明电子邮件已成功发送到多个收件人的邮箱。

图 2.41　群发普通文本邮件

关键技术

应用 Spring 的 E-mail 抽象层群发普通的邮件，主要用到 SimpleMailMessage 类，该类用于表示普通的文本邮件对象。在实现群发时，主要用到 SimpleMailMessage 对象的 setTo(String [] to)方法，该方法用于设置多个收件人的地址。

（1）创建发送邮件的表单页 index.jsp，在该页中添加表单元素，用于发送普通的邮件。具体代码参见配书

光盘。

（2）创建 MessageInfo 类，用于封装邮件信息。关键代码如下：

```
public class MessageInfo {
        private String serverHost="";                           //邮件服务器
        private String password;                                //邮箱密码
        private String from = "";                               //发件人地址
        private String [] to;                                   //收件人地址数组
        private Date sendDate;                                  //发送时间
        private String subject;                                 //邮件主题
        private String msg="";                                  //消息正文
        private String bcc = "";                                //暗送邮件地址
        private String cc="";                                   //抄送邮件地址
        ……//省略了 getter 和 setter 方法
}
```

（3）自定义发送邮件的类 EmailUtil，在该类中创建一个用于依赖注入的 JavaMailSender 属性，并编写一个 doSend(MessageInfo msg)方法，实现邮件的群发。关键代码如下：

```
public class EmailUtil {
 private JavaMailSender mailSender;                             //注入 Spring E-mail 抽象层的发送邮件对象
 public JavaMailSender getMailSender() {
        return mailSender;
 }
 public void setMailSender(JavaMailSender mailSender) {
        this.mailSender = mailSender;
 }
 public    void doSend(MessageInfo msg){
        SimpleMailMessage message = new SimpleMailMessage();    //创建邮件对象
        message.setFrom(msg.getFrom());                         //设置发送者地址
        message.setTo(msg.getTo());                             //设置多个接收者地址
        message.setSubject(msg.getSubject());                   //设置主题
        message.setSentDate(msg.getSendDate());                 //设置发送时间
        message.setText(msg.getMsg());                          //设置消息内容
        JavaMailSenderImpl sender = (JavaMailSenderImpl)mailSender;   //邮件发送对象
        sender.setHost(msg.getServerHost());                    //设置邮件主机地址
        sender.setUsername(msg.getFrom());                      //设置邮箱用户名
        sender.setPassword(msg.getPassword());                  //设置密码
        sender.send(message);                                   //发送邮件
 }
}
```

（4）创建 Spring 的配置文件 applicationContext.xml，对 JavaMailSender 进行初始化配置，并注入到 EmailUtil 对象中。具体代码参见配书光盘。

（5）创建处理邮件表单页面 mydeal.jsp，在该页中获取表单中的邮件信息，然后装载 Spring 的配置文件，应用 Spring 的 BeanFactory 获取 EmailUtil 对象，并调用 EmailUtil 对象的 doSend()方法实现群发普通文本邮件。具体代码参见配书光盘。

心法领悟 061：邮件群发。

在 Spring 的 E-mail 抽象层中，用 SimpleMailMessage 表示普通邮件对象，用 MimeMailMessage 表示复合的（可以添加 HTML 格式或附件）邮件对象，它们都继承自 MailMessage 接口，并且都实现了 MailMessage 接口的 setTo(String [] to)方法，所以只要通过相应的邮件对象的 setTo(String [] to)方法设置多个接收者的邮件地址即可实现邮件群发。

第 2 篇

数据库应用篇

⏭ 第 3 章　数据库操作技术

⏭ 第 4 章　SQL 语句应用技术

⏭ 第 5 章　复杂查询技术

⏭ 第 6 章　数据库高级应用

第 3 章

数据库操作技术

- 建立 Connection 数据库连接
- 数据库与数据表
- 数据库的增加、删除与更新操作

3.1　建立 Connection 数据库连接

实例 062	建立 Access 数据库连接 光盘位置：光盘\MR\03\062	中级 实用指数：★★☆

实例说明

Access 作为关系型桌面数据库管理系统，在建立中小型数据库管理系统中得到了广泛的应用。通过 JDBC-ODBC 桥连接数据库是一种很简单的方法，无须在项目中添加驱动文件。本实例实现的是通过 JDBC-ODBC 桥连接 Access 数据库，运行结果如图 3.1 所示。

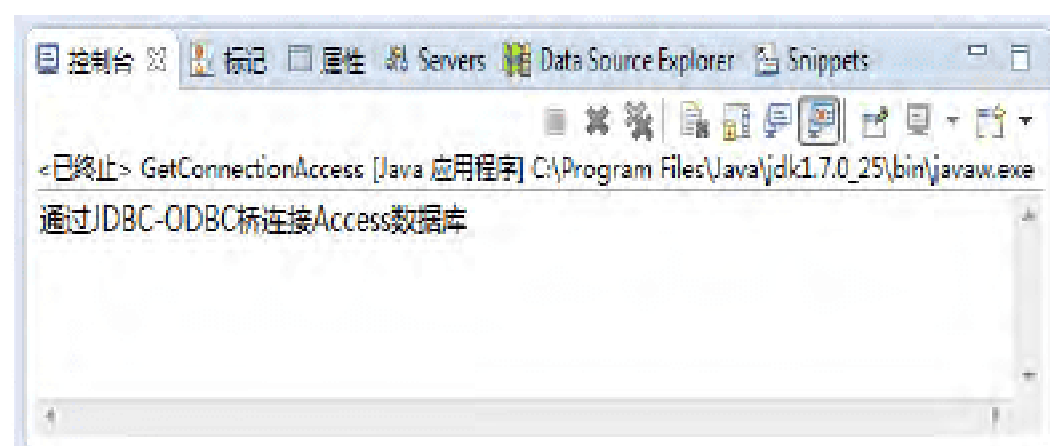


图 3.1　建立 Access 数据库连接

关键技术

ODBC 是一种访问数据库的方法，只要系统中有相应的 ODBC 驱动程序，任何程序都可以通过 ODBC 驱动程序操纵数据库。

在使用 ODBC 时，经常提到 DSN（Data Source Name，数据源名）这个名词。在给 ODBC 驱动程序传递 SQL 指令时，通过 DSN 来告诉 ODBC 驱动程序到底操作哪一个数据库。如果数据库的平台发生改变，例如，改为 SQL Server 数据库，只要其中表的结构没变，就无须改写程序，只要重新在系统中配置 DSN 即可。

由此可见，DSN 是应用程序和数据库之间的桥梁，要通过 ODBC 访问数据库，前提必须配置好 DSN，即为 DSN 指定一个名称，而这个名称的作用就是通知系统调用哪个 ODBC 驱动程序。

（1）配置 Microsoft Access 数据库文件的 DSN。在“控件面板”窗口中双击“管理工具”/“数据源（ODBC）”，打开如图 3.2 所示的“ODBC 数据源管理器”对话框。

（2）选择“系统 DSN”选项卡，单击“添加”按钮，打开如图 3.3 所示的“创建新数据源”对话框。

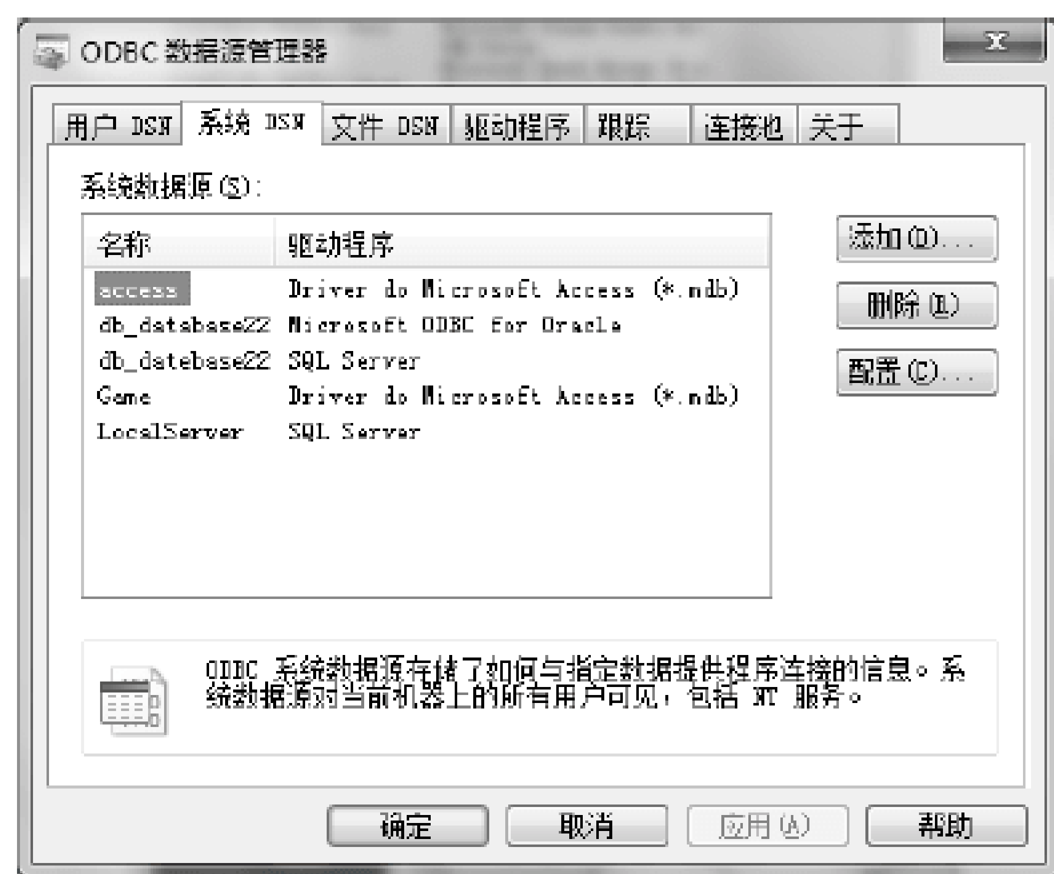


图 3.2　“ODBC 数据源管理器”对话框



图 3.3　“创建新数据源”对话框

（3）从“选择您想为其安装数据源的驱动程序”列表框中选择 Microsoft Access Driver 选项，然后单击“完成”按钮，打开如图 3.4 所示的“ODBC Microsoft Access 安装”对话框。

（4）在“数据源名”文本框中输入数据源名称 Access，然后单击“选择”按钮，在弹出的如图 3.5 所示的“选择数据库”对话框中选择要和数据源连接的数据库，单击“确定”按钮。



图 3.4 “ODBC Microsoft Access 安装”对话框

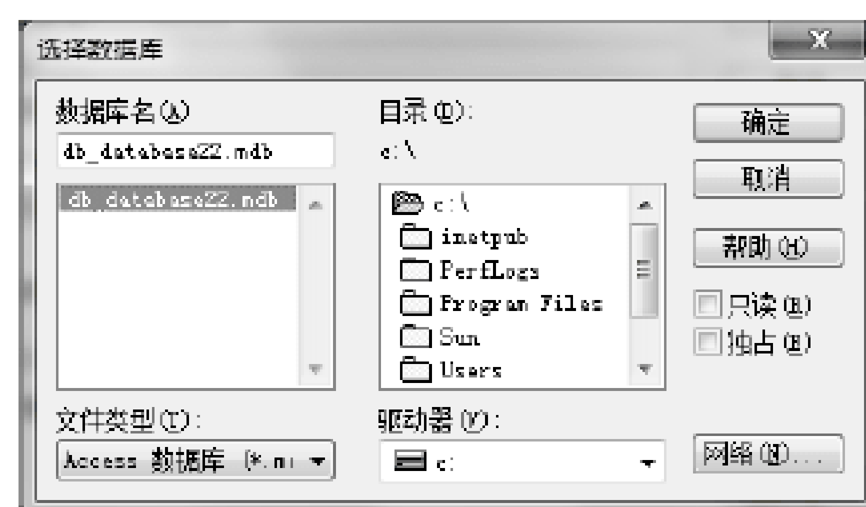


图 3.5 “选择数据库”对话框

（5）单击“确定”按钮，完成 Microsoft Access 数据库文件 DSN 的配置工作。

（6）数据库创建成功后，可以通过 Java 应用程序测试数据源是否连接成功。在项目中创建 Java 类 GetConnectionAccess，该类中定义验证数据库连接方法。具体代码如下：

```
public boolean Connection(){
    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");                    //加载数据库驱动
        Connection con = DriverManager.getConnection("jdbc:odbc:access");  //获取数据库连接
        if(con != null){
            System.out.println("通过 JDBC-ODBC 桥连接 Access 数据库");
        }
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

秘笈心法

心法领悟 062：理解抽象类为什么不能进行实例化。

理解这个问题，其实很简单，因为在抽象类中包含着未实现的抽象方法，如果产生了抽象类的对象，那么用户要使用对象调用这些方法该怎么办？因此抽象类是不能创建对象的。实现抽象类中的抽象方法，交给其子类完成，如果某个类继承了一个抽象类，但没实现其抽象方法时，这个类也必须被定义为抽象类。

实例 063 建立与 MySQL 数据库的连接

光盘位置：光盘\MR\03\063

中级　实用指数：★★☆

实例说明

操作数据库的第 1 步就是要获取程序与数据库的连接。本实例实现与 MySQL 数据库建立连接，并将数据库中的表 tb_bccd 中的数据显示在页面中，如图 3.6 所示。

图 3.6 建立与 MySQL 数据库的连接

关键技术

目前，MySQL 的 JDBC 包主要有 Jconnector 和 org.git.mm.mysql，下面分别介绍。

- Jconnector 包：Jconnector 包是 MySQL 官方网站公布的，其更新速度比较快，很多程序员都使用该包。
- org.git.mm.mysql 包：org.git.mm.mysql 包是国外一些 Java 爱好者编写的，出现的时间比较长，国际化程度做得比较好，而且对中文支持也比较好。本实例使用的就是该包。

连接 MySQL 数据库的驱动程序，代码如下：

```
org.gjt.mm.mysql.Driver
```

URL 地址的代码如下：

```
jdbc:mysql://IP:PORT/databaseName?user=UserName&password=PWD&useUnicode=true
```

参数说明

❶ IP：MySQL 主机的 IP 地址。

❷ PORT：MySQL 主机的端口号，3306 为安装 MySQL 时的默认端口号。

❸ useUnicode：用于设置是否使用 Unicode 输出。

设计过程

创建 Web 项目，在该项目中定义 GetConn 类，在该类中定义与 db_database03 数据库获取连接的方法 getConnection()。关键代码如下：

```
public Connection getConnection() {
    try {
        Class.forName("com.mysql.jdbc.Driver");                          //加载 MySQL 数据库驱动
        System.out.println("数据库驱动加载成功！！");
        String url = "jdbc:mysql://localhost:3306/db_database03";        //定义连接数据库的 URL
        String user = "root";                                            //定义连接数据库的用户名
        String passWord = "111";                                         //定义连接数据库的密码
        conn = DriverManager.getConnection(url, user, passWord);         //获取数据库连接
        System.out.println("已成功的与 MySQL 数据库建立连接！！");
    } catch (Exception e) {
        e.printStackTrace();
    }
    return conn;
}
```

秘笈心法

心法领悟 063：显示查询结果。

在使用表格显示查询结果时，通常会创建两行的表格，第 1 行显示表头信息，在第 2 行的开始位置处使用循环创建表格的行与列，来显示查询结果。

实例 064　建立与 SQL Server 2000 数据库的连接

光盘位置：光盘\MR\03\064

中级　实用指数：★★☆

实例说明

要实现 Java 程序与数据库的连接，必须使用数据库厂商提供的数据库驱动。数据库不同，驱动也不同，因此连接数据库的代码也不同。本实例将实现 Java 程序与 SQL Server 2000 数据库建立连接，连接成功后在 Eclipse 控制台上输出提示信息，如图 3.7 所示。



图 3.7　建立与 SQL Server 2000 数据库的连接

关键技术

本实例使用 jtds 驱动包建立 SQL Server 2000 数据库与 Java 程序的连接。建立正确的数据库连接，必须保证连接数据库驱动的代码与连接数据库的 URL 书写正确。通过 Class 类的 forName()方法实现注册 SQL Server 数据库驱动类，关键代码如下：

```
Class.forName("net.sourceforge.jtds.jdbc.Driver");            //加载数据库驱动
```

java.sql 包中的 Connection 接口代表数据库的连接，使用 DriverManager 类的 getConnection()方法可获取数据库的连接。在 SQL Server 2000 数据库中需要指定连接数据库的 URL、用户名、密码。本实例连接数据库的

URL 代码如下：

```
String url = "jdbc:jtds:sqlserver://localhost:1433;DatabaseName=master";
```

连接数据库采用的用户名为 sa，密码为空。

设计过程

（1）创建 Java 项目，在该项目中创建类 GetConn，在该类的静态块中实现加载数据库驱动。关键代码如下：

```
static {
    try {
        Class.forName("net.sourceforge.jtds.jdbc.Driver");                 //加载数据库驱动
        System.out.println("数据库驱动加载成功！");
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}
```

（2）在该类中定义 getConn()方法，获取与数据库的连接，关键代码如下：

```
public Connection getConn() {
    String url = "jdbc:jtds:sqlserver://localhost:1433;DatabaseName=master";   //连接数据库的 URL
    String userName = "sa";                                                    //连接数据库的用户名
    String passWord = "";                                                      //连接数据库的密码
    try {
        conn = DriverManager.getConnection(url, userName, passWord);           //获取数据库连接
        if (conn != null) {
            System.out.println("已成功的与 SQLServer2000 数据库建立连接！");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return conn;                                                               //返回 Connection 对象
}
```

秘笈心法

心法领悟 064：mssqlserver.jar、msutil.jar、msbase.jar 驱动包。

在本实例中，与数据库建立连接使用的是 jtds.jar 驱动包。除此之外，也可以使用其他驱动包，即 mssqlserver.jar、msutil.jar、msbase.jar。当然，使用这 3 个驱动包时，加载数据库驱动的代码需要改变，代码如下：

```
Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
```

实例 065 建立与 SQL Server 2005 数据库的连接

光盘位置：光盘\MR\03\065

中级

实用指数：★★☆

实例说明

相对于 SQL Server 2000 数据库，SQL Server 2005 数据库有了很大的改进，因此建立与 SQL Server 2005 数据库的连接与 SQL Server 2000 数据库有所区别，采用的驱动也不一样。本实例将实现通过 JDBC 技术建立 Java 程序与 SQL Server 2005 数据库的连接，运行结果如图 3.8 所示。

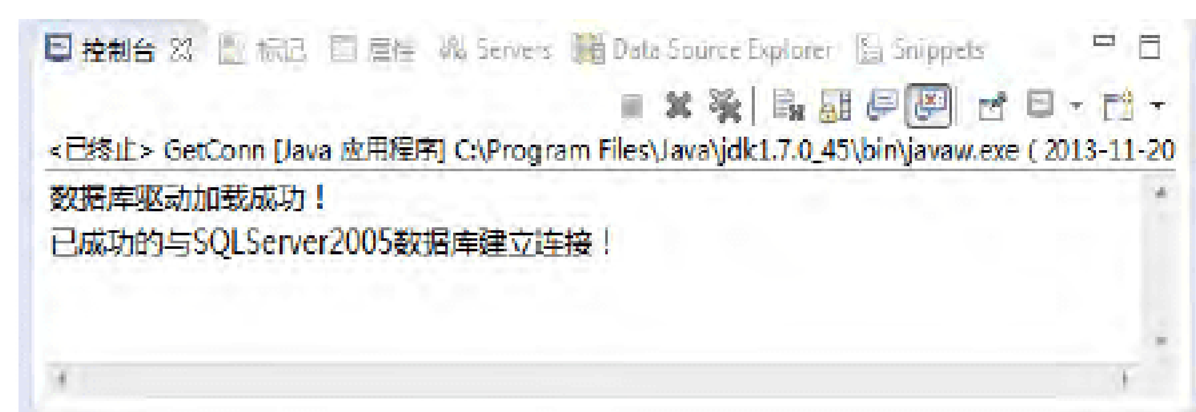


图 3.8　建立与 SQL Server 2005 数据库的连接

连接 SQL Server 2005 数据库应用的驱动程序是 sqljdbc.jar，可以到 Microsoft 的官方网站（网址为 http://www.microsoft.com）下载。在此需注意的是，很多初学者会使用连接 SQL Server 2000 数据库的代码来连接 SQL Server 2005，结果导致一些问题。其实连接 SQL Server 2000 与连接 SQL Server 2005 的驱动与 URL 有一些差别，如表 3.1 所示。

表 3.1　连接 SQL Server 2000 与 SQL Server 2005 的区别

数　据　库	驱　　动	URL
SQL Server 2000	com.microsoft.jdbc.sqlserver.SQLServerDriver	jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=数据库名称
SQL Server 2005	com.microsoft.sqlserver.jdbc.SQLServerDriver	Jdbc:sqlserver://localhost:1433;DatabaseName=数据库名称

设计过程

在项目中创建类 GetConn，在该类中定义 getConnection()方法，用于建立与数据库的连接。具体代码如下：

```
public Connection getConnection(){                                          //定义连接数据库方法
    try {
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");      //加载数据库驱动
        System.out.println("数据库驱动加载成功！");
        String url = "jdbc:sqlserver://localhost:1433;DatabaseName=master"; //定义连接数据库 URL
        String userName = "sa";
        String passWord = "";
        conn = DriverManager.getConnection(url,userName ,passWord);        //获取数据库连接
        if(conn != null){
            System.out.println("已成功的与 SQLServer2005 数据库建立连接！");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return conn;
}
```

秘笈心法

心法领悟 065：常见连接问题。

在数据库连接过程中，可能会出现一些连接异常问题。要保证与 SQL Server 2005 数据库的连接，必须保证开启 TCP/IP 服务，设置 TCP 端口为 1433，因为 SQL 服务器默认端口是没有配置的，所以要重新设置。

实例 066	建立与 Oracle 数据库的连接 光盘位置：光盘\MR\03\066	中级 实用指数：★★☆

实例说明

Oracle 数据库的数据管理功能十分强大，对计算机的要求比较高。由于该数据库在 IT 行业中占据着相当重要的地位，程序员必须学会使用。使用 Oracle 数据库的前提是与数据库建立连接。本实例将应用 JDBC 技术连接 Oracle 数据库，运行结果如图 3.9 所示。

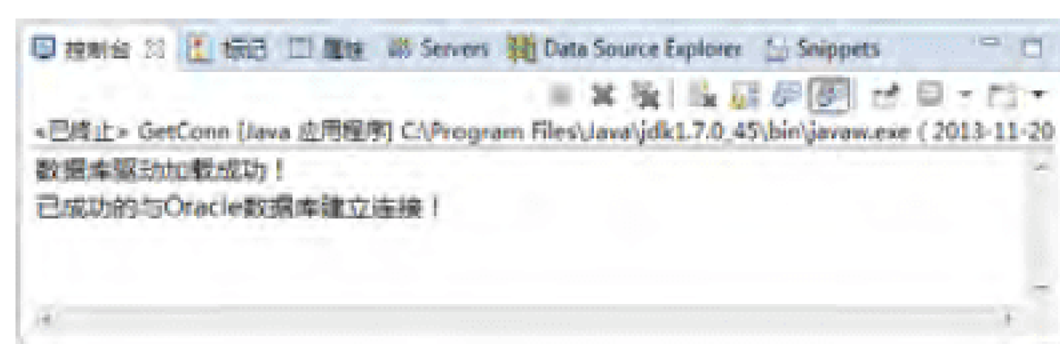


图 3.9　建立与 Oracle 数据库的连接

通过 JDBC 连接 Oracle 数据库仍然可分为两步：加载数据库驱动；获取数据库连接。本实例连接 Oracle 数据库应用的驱动是 classes12.jar。加载数据库驱动代码如下：

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

在获取数据库连接时，需要定义连接数据库的 URL。在此要注意的是，连接 Oracle 数据库与连接 SQL Server 数据库的 URL 存在较大的差异。本实例获取数据库连接的 URL 地址，代码如下：

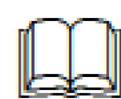
```
String url="jdbc:oracle:thin:@localhost:1521:orcl3";
```

参数说明

❶ @：分隔符。

❷ 1521：数据库端口。

❸ orcl3：数据库名或 SID。

 说明：SID 是数据库的唯一标识符，是在建立一个数据库时系统自动赋予的一个初始 ID。

设计过程

在项目中创建 GetConn 类，在该类中创建 getConnection()方法，获取与数据库的连接。具体代码如下：

```
public Connection getConnection() {
    Connection conn = null;
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");                    //加载数据库驱动
        System.out.println("数据库驱动加载成功！ ");                          //输出的信息
        String url = "jdbc:oracle:thin:@localhost:1521:orcl3";               //获取连接 URL
        String user = "system";                                              //连接用户名
        String password = "aaa";                                             //连接密码
        Connection con = DriverManager.getConnection(url, user, password);   //获取数据库连接
        if (con != null) {
            System.out.println("成功的与 Oracle 数据库建立连接！ ");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return conn;                                                             //返回 Connection 实例
}
```

秘笈心法

心法领悟 066：SID 的用处。

SID 和数据库名都是数据库的唯一标识，但是在作用上却有很大的差异。SID 主要用于一些 DBA 操作以及与操作系统进行交互，例如，从操作系统的角度访问实例名，必须通过 Oracle SID（操作系统的环境变量）。此外，SID 在注册表中也存在。在安装数据库、创建新的数据库、创建数据库控制文件、修改数据结构、备份与恢复数据库时，都需要用到数据库名。



实例 067 建立与 Java DB 数据库的连接

光盘位置：光盘\MR\03\067

中级 实用指数：★★☆

实例说明

Java DB 是在安装 JDK 时自动安装的，不需要另外安装数据库系统，使用起来很简单，尤其对于小型应用程序使用 Java DB 数据库非常方便，但 Java DB 数据库并没有提供企业管理器等用户交互界面，在使用该数据库时需要注意这一点。本实例实现的是通过 Java 程序连接 Java DB 数据库。在连接的过程中需要注意，如果连接的数据库不存在，需要通过程序创建相应的数据库；如果连接成功，将给出如图 3.10 所示的运行结果。

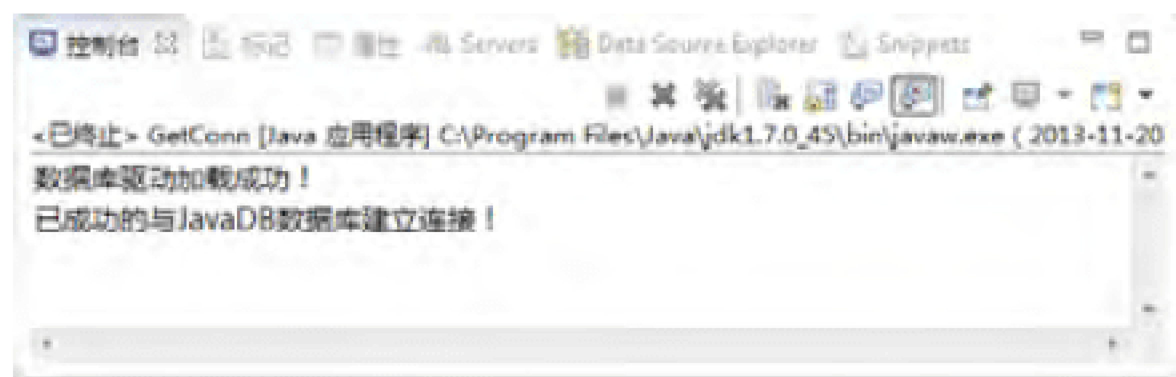


图 3.10　建立与 Java DB 数据库的连接

关键技术

连接 Java DB 数据库需要的驱动为 derby.jar，读者可到网上下载使用。连接 Java DB 数据库与连接其他类型的数据库方法相同。具体参见前文所述其他数据库的连接。

在项目中创建类 JavaDBConnection，在该类中的静态块中定义代码，获取数据库连接。代码如下：

```
private static final String DRIVERCLASS = "org.apache.derby.jdbc.EmbeddedDriver";              //数据库驱动
private static final String URL = "jdbc:derby:db_database03";                                  //数据库 URL
private static final ThreadLocal<Connection> threadLocal = new ThreadLocal<Connection>();      //创建用来保存数据库连接的线程
```

```
private static Connection conn = null;
static {                                    //通过静态方法加载数据库驱动，并且在数据库不存在的情况下创建数据库
    try {
        Class.forName(DRIVERCLASS);                                    //加载数据库驱动
        System.out.println("数据库驱动加载成功！");
        File albumF = new File("db_database03");                       //创建数据库文件对象
        conn = DriverManager.getConnection(URL + ";create=true");      //创建数据库连接
        System.out.println("已成功的与 JavaDB 数据库建立连接！");
        threadLocal.set(conn);                                         //保存数据库连接
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 067：及时关闭数据库连接。

完成与数据库的交互后，要及时关闭与数据库的连接，释放系统占用的资源。虽然 JVM 会定时清理缓存，但当数据库连接达到一定数量时，若清理不够及时，将严重影响数据库和计算机的运行速度。

3.2　数据库与数据表

实例 068　列举 SQL Server 数据库中的数据表

光盘位置：光盘\MR\03\068

中级　实用指数：★★☆

实例说明

本实例实现获取数据库 db_database03 中的所有数据表（可通过 java.sql 包中的 DatabaseMetaData 接口来获取），并将查询得到的数据表名显示在页面中，如图 3.11 所示。

列举数据库中的所有数据表	
编号	数据表
1	dtproperties
2	tb_bccd
3	tb_dept
4	tb_emp
5	tb_stu
6	tb_teacher
7	tb_use

图 3.11　列举数据库中的所有数据表

DatabaseMetaData 接口由驱动程序供应商实现，用户可以从中了解 Database Management System（DBMS）在与驱动程序相结合时的能力。不同的关系数据库管理系统常常支持不同的功能，以不同方式实现这些功能，并使用不同的数据类型。此外，驱动程序可以实现 DBMS 提供的顶级功能。

使用该接口中的 getTables()方法，可获取指定数据库中的所有数据表名。getTables()方法返回 ResultSet 数据结果集。具体语法如下。

```
getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)
```

该方法的参数说明如表 3.2 所示。

表 3.2　getTables()方法参数说明

参　　数	类　　型	描　　述
catalog	String	类别名称，必须与存储在数据库中的类别名称匹配
schemaPattern	String	模式名称，必须与存储在数据库中的模式名称匹配
tableNamePattern	String	表名称模式，必须与存储在数据库中的表名称匹配
types	String[]	要包括的表类型所组成的列表

设计过程

创建 Web 项目，在该项目中创建用于操作数据的类 JDBCDao，在该类中定义 getRs()方法，用于获取数据库中的所有数据表，具体代码如下：

```
public List getRs() {
	try {
		List list = new ArrayList();
		String[] tableType = { "TABLE" };                                      //指定要进行查询的表类型
		conn = getConn();                                                       //调用与数据库建立连接方法
		DatabaseMetaData databaseMetaData = conn.getMetaData();                 //获取 DatabaseMetaData 实例
		ResultSet resultSet = databaseMetaData.getTables(null, null, "%",
				tableType);                                                     //获取数据库中所有数据表集合
		while (resultSet.next()) {
			list.add(resultSet.getString("TABLE_NAME"));
		}
		return list;
	} catch (SQLException e) {
		System.out.println("记录数量获取失败！ ");
		return null;
	}
}
```

说明： getTables()方法获取的 ResultSet 数据库集合表由多个列组成，其中列名称为 TABLE_NAME 的数据列，用来存储数据库中的所有数据表集合。

秘笈心法

心法领悟 068：合理运用 SQL Server 数据库中的系统表。

本实例实现的是将数据库中的用户表检索出来。除此之外，在 SQL Server 数据库中还包含一些系统表，下面分别介绍。

- 系统表 sysobjects：用于记录在数据库内创建的每个对象（约束、默认值、日志、规则、存储过程等）。该表中的 name 字段记录了所有对象的名称。
- 系统表 sysprocesses：用于保存运行在 Microsoft® SQL Server™ 上的进程信息，这些进程可以是客户端进程或系统进程。该表中的 smallid 字段记录了所有表的字段 ID 号；value 字段记录了所有表字段的描述信息。
- 系统表 syscolumns：用于记录每个表和视图中的每列，以及存储过程中的每个参数。该表位于每个数据库中。该表中的 name 字段记录了所有表的记录名。

实例 069 列举 MySQL 数据库中的数据表

光盘位置：光盘\MR\03\069

中级

实用指数：★★☆

实例说明

MySQL 数据库提供了 SHOW TABLES 子句用来获取数据表。本实例将实现获取 MySQL 数据库中的数据表，并将其显示在页面中，如图 3.12 所示。

列举MySQL数据库中的所有数据表	
编号	数据表
1	tb_bccd
2	tb_stu

图 3.12　列举 MySQL 数据库中的所有数据表

在 MySQL 中通过 SHOW 语句可以列出指定数据库中的数据表。

语法：

```
SHOW TABLES [FROM databaseName] [LIKE expression];
```

参数说明

❶ databaseName 子句：可选项，用于指定要获取数据表的数据库。省略该子句，则列举当前打开数据库中的数据表；如果当前没有打开数据库，则返回错误信息。

❷ expression 子句：可选项，用于设置列举条件。expression 是一个字符型表达式，可以包括通配符，如百分号（代表多个字符）、下划线（代表一个字符）等。例如，LIKE '%book%'将获取名称中包括 book 的数据表。

注意：SHOW TABLES 是 MySQL 数据库特有的方法，在其他数据库中无效。

设计过程

（1）创建项目，在该项目中创建 JdbcUtil 类，在该类中定义连接数据库的方法 getConnection()。具体代码参见配书光盘中的源程序。

（2）在 JdbcUtil 类中定义 listDB()方法，用于获取数据库中所有数据表。具体代码如下：

```
public List listDB() {
        List list = new ArrayList();
    String sql = "show tables;";                                    //定义查询数据 SQL 语句
    try {
        conn = getConnection();                                     //获取数据库连接
        Statement stmt = conn.createStatement(
                ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_READ_ONLY);                        //实例化 Statement 对象
        ResultSet rs = stmt.executeQuery(sql);                      //执行查询 SQL 语句
        while(rs.next()){
            list.add(rs.getString(1));
        }
        return list;                                                //返回查询结果
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
        return null;
    }
}
```

秘笈心法

心法领悟 069：不要将数据库特有的函数应用在其他数据库中。

SQL 语句对于数据库来说是通用的，但是有些数据库特有的函数是不能用在其他数据库中的。例如，SQL Server 数据库中的 top 关键字、MySQL 数据库中的 limit 函数，还有本实例介绍的 show tables 语句。

实例 070　查看数据表结构

光盘位置：光盘\MR\03\070

高级

实用指数：★★★

实例说明

数据表的结构包括表中的字段名、字段类型、字段长度等属性。本实例实现的是查询 SQL Server 数据库中的数据表结果，并将查询结果在页面中显示，如图 3.13 所示。

显示数据表结构

选择数据表：dtproperties　查询

字段编号	字段名	字段长度	字段类型	描述	是否为空	小数位数	默认值
1	id	4	int			0	
2	name	10	char				

图 3.13　查看数据表结构

本实例主要应用 SQL Server 系统表 Sysobjects（系统对象）、Syscolumns（字段）、Sysproperties（描述）、Systypes（字段类型）和 Syscomments（默认值）等相对应的表关系显示出表的结构，这些系统表都可以通过对象编号

和相关表编号进行关联。

设计过程

在项目中创建类 JdbcUtil，用于定义操作数据库的相关方法（获取数据库连接的代码参见配书光盘中的源程序）；然后在该类中定义查询指定数据表结构的方法 getMessage()（该方法以 String 类型变量为参数，用于指定要查询的数据库）。具体代码如下：

```
public List getMessage(String tableName) {
        List list = new ArrayList();                                          //定义保存返回值的 List 集合
        String SQL = " Select case when c.colid=1 then    o.name end  表名,"
                    + " c.ColId  字段编号,c.name  字段名,c.length  字段长度,t.name  字段类型,"
                    + " p.value  描述,case when c.isnullable=0 then '1' end  是否为空,"
                    + " c.scale  小数位数,REPLACE (REPLACE (REPLACE (m.text,'(',''),')',''),'''','')  默认值,"
                    + " case when ("
                    + " Select Count(*) From SysObjects where name in ("
                    + " Select name From Sysindexes Where id=c.id and indid in ("
                    + " Select indid From Sysindexkeys    where id=c.id and colid in ("
                    + " Select colid From Syscolumns where id=c.id and colid=c.colid))) and xtype='pk')>0"
                    + " then '1' end  是否为主键"
                    + " From Sysobjects o"
                    + " left join Syscolumns c on o.id=c.id"
                    + " left join Sysproperties p on o.id=p.id and c.colid=p.smallid"
                    + " left join Systypes t on t.xtype=c.xtype"
                    + " left join Syscomments m on m.id=c.cdefault"
                    + " where (o.xtype='u' or o.xtype='v') and o.status>0 and o.name='"
                    + tableName + "'" + " order by o.name,c.colid";           //定义查询 SQL 语句
        ResultSet res = GetRs(SQL);                                           //调用执行 SQL 语句方法
        ResultSetMetaData Rsmd;                                               //获取 ResultSetMetaData 方法
        try {
                Rsmd = res.getMetaData();                                     //实例化 ResultSetMetaData 对象
                while (res.next()) {                                          //循环遍历查询结果集
                        Student student = new Student();                      //创建对应数据库对象的 JavaBean 对象
                        student.setId(res.getString("字段编号"));              //设置对象属性
                        student.setName(res.getString("字段名"));
                        student.setType(res.getString("字段类型"));
                        student.setAcquiescence(res.getString("默认值"));
                        student.setDepict(res.getString("描述"));
                        student.setDigit(res.getString("小数位数"));
                        student.setLength(res.getString("字段长度"));
                        student.setIfNull(res.getString("是否为空"));
                        list.add(student);                                    //将对象添加到 List 集合中
                }
        } catch (SQLException e) {
                e.printStackTrace();
        }
        return list;                                                          //返回 List 集合
}
```

秘笈心法

心法领悟 070：端口连接异常。

安装 SQL Server 2000 数据库后，可能在获取数据库与 Java 程序连接时报出数据库连接异常。这样的问题可能是由于没有安装补丁所致。SQL Server 2000 数据库常用的补丁有两种，即 sp3、sp4。如果没有安装补丁，连接时就会报出连接端口异常的错误提示。

实例 071	动态维护投票数据库 光盘位置：光盘\MR\03\071	高级 实用指数：★★★★

实例说明

本实例实现的是投票选出用户心中最好的图书，运行结果如图 3.14 所示。用户可以通过程序在数据表中添

加一列，也可以从数据表中删除一列，进而实现投票统计功能。

图 3.14　动态维护搜索数据库

关键技术

本实例主要使用 ALTER TABLE 语句中的 ADD 和 DROP 子句来实现。

（1）ADD 子句

此子句主要用于向数据表中添加字段。

语法：

```
ALTER TABLE table
ADD [ < column_definition > ] | column_name AS computed_column_expression
```

参数说明

❶ table：要添加字段的数据表名称。

❷ column_definition：字段的定义。

❸ column_name：字段的名称。

❹ computed_column_expression：计算字段的表达式。

（2）DROP 子句

此子句主要用于删除数据表中的字段。

语法：

```
ALTER TABLE table
DROP constraint_name
```

参数说明

❶ table：需要删除字段的数据表名。

❷ constraint_name：需要删除字段的名称。

（1）创建 UserDao 类，定义连接、查询和关闭数据库的方法。

（2）创建 index.jsp 页面。首先，通过 JavaBean 标签调用 UserDao 类，添加 form 表单，执行 UserDao 类中的 selectStatic()方法，将查询结果集中的投票选项作为单选按钮的值；然后，创建文本框，用于提交投票的备选答案；最后，将查询结果集中的投票选项输出到下拉列表框中，用于实现删除指定投票选项。这里将文本框、下拉列表和单选按钮的值都提交到 Create_DataBase 类中。关键代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java" import="java.sql.*" errorPage="" %>
<jsp:useBean id="dao" class="com.pkh.dao.UserDao" scope="page"/>
<form name="form1" method="post" action="Create_DataBase">
<%
    ResultSet Rs = dao.selectStatic("Select Top 1 * From tb_Tou");          //执行查询投票选项的操作
    ResultSetMetaData Rsmd = Rs.getMetaData();                              //获取查询结果
    for (int i=2;i<=Rsmd.getColumnCount();i++){                             //循环输出查询结果
%>
<tr>
        <td width="115"><div align="center"><%=Rsmd.getColumnName(i) %></div></td>
```

```
        <td width="117"><div align="left">
          <input type="radio" name="radiobutton" value="<%=Rsmd.getColumnName(i) %>">
          </div></td>
</tr>
<%    }        %>
<tr>
          <td width="113" height="23" nowrap><div align="right">添加备选答案：</div></td>
        <td width="131"><div align="left">
          <input name="tadd" type="text" id="tadd" size="15">
          </div></td>
        <td><input name="Submit" type="submit" id="Submit" value="添加"></td>
</tr>
<tr>
          <td height="23"><div align="right">删除备选答案：</div></td>
        <td><div align="left">
                  <select name="sdelete" id="sdelete">
             <%
                          for (int i=2;i<=Rsmd.getColumnCount();i++){   //将查询结果循环输出到下拉列表框中
                  %>
                      <option value="<%=Rsmd.getColumnName(i) %>"><%=Rsmd.getColumnName(i) %></option>
                  <%    }          %>
                  </select>
                  </div></td>
                  <td><input name="Submit" type="submit" id="Submit" value="删除"></td>
          </tr>
</form>
```

（3）创建 Servlet 类 Create_DataBase，定义 doPost()方法，完成投票选项的添加、修改和删除操作。关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        UserDao dao = new UserDao();
        String btn = request.getParameter("Submit");
        String SQL = "";
        if (btn.equals("投票")) {                                           //执行投票添加的操作
                String tou = request.getParameter("radiobutton");
                if (tou.length() > 0) {
                        SQL = "insert tb_Tou (" + tou + ") values(1)";             //编写添加语句
                        dao.CreateDataBase(SQL);
                        out.println("<script >alert('投票成功！'); window.location.href='index.jsp';</script>");
                }
        } else if (btn.equals("添加")) {                                    //添加投票选项
                String add = request.getParameter("tadd");
                if (add.length() > 0) {
                        SQL = "ALTER TABLE tb_Tou ADD " + add + " bit ";           //编写添加语句
                        dao.CreateDataBase(SQL);
                        out.println("<script >alert('创建成功！'); window.location.href='index.jsp';</script>");
                }
        } else if (btn.equals("删除")) {                                    //删除投票选项
                String del = request.getParameter("sdelete");
                if (del.length() > 0) {
                        SQL = "ALTER TABLE tb_Tou DROP COLUMN " + del;             //编写删除语句
                        dao.CreateDataBase(SQL);
                        out.println("<script >alert('删除成功！'); window.location.href='index.jsp';</script>");
                }
        }
}
```

（4）创建 search.jsp 页面，输出投票结果。

```
<%@ page contentType="text/html; charset=gbk" language="java" import="java.sql.*" errorPage="" %>
<jsp:useBean id="dao" class="com.pkh.dao.UserDao" scope="page"/>
<%
        ResultSet Rs = dao.selectStatic("Select top 1 * From tb_Tou");
        ResultSetMetaData Rsmd = Rs.getMetaData();
      for (int i=2;i<=Rsmd.getColumnCount();i++){
%>
<tr>
```

```
        <td width="85"><div align="center"><%=Rsmd.getColumnName(i) %></div></td>
<%
        ResultSet RSCount = dao.selectStatic("Select count(*) Count From tb_Tou Where "+Rsmd.getColumnName(i)+"=1");
        RSCount.next();
%>
<td width="129"><div align="center"><%=RSCount.getString(1) %></div></td>
```

秘笈心法

心法领悟 071：浏览器怎样进行 URL 编码。

浏览器在显示网页文档时，需要使用某些字符集编码来显示其中的内容。对于 FORM 表单中输入的中文字字符，浏览器将按照当前显示页面所采用的字符集编码来进行 URL 编码，即浏览器先将 FORM 表单中输入的内容转换成当前显示网页所选择的字符集编码，再对这些内容进行 URL 编码，最后传送给 Web 服务器。

实例 072	SQL Server 数据库的备份 光盘位置：光盘\MR\03\072	中级 实用指数：★★✩

实例说明

对于一个大型网站来说，数据恢复功能至关重要。利用该功能，可以在数据遭到破坏时，将备份的数据恢复到系统中，保证系统重新正常运转，从而避免因数据异常丢失所带来的损失。为了使用户所开发的网站在数据处理方面具有更高的安全性，建议在网站开发时设计数据备份及数据恢复功能。本实例实现的是将 SQL Server 数据库进行备份，并将备份后的文件保存在 C 盘。本实例的运行结果如图 3.15 所示。

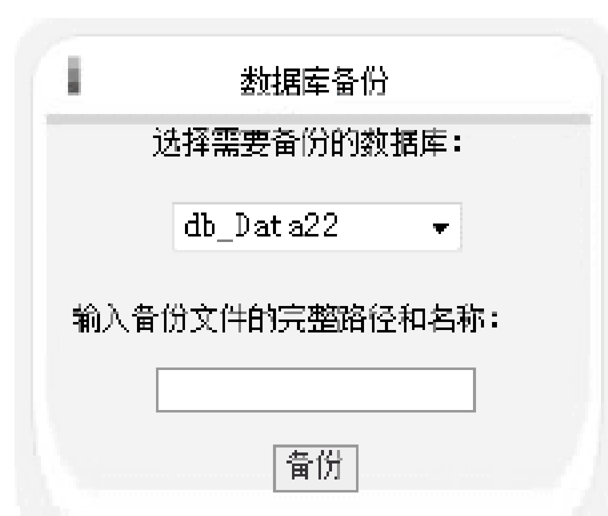


图 3.15　数据库备份

本实例运用 SQLDMO.backup 对象完成整个系统数据库的备份，以便在系统或数据库发生故障（例如，硬盘发生故障）时可以重建系统。

备份整个数据库的语法如下。

```
BACKUP DATABASE { database_name | @database_name_var }
TO < backup_device > [ ,...n ]
[ WITH
    [ BLOCKSIZE = { blocksize | @blocksize_variable } ]
    [ [ , ] DESCRIPTION = { 'text' | @text_variable } ]
    [ [ , ] DIFFERENTIAL ]
    [ [ , ] EXPIREDATE = { date | @date_var }
        | RETAINDAYS = { days | @days_var } ]
    [ [ , ] PASSWORD = { password | @password_variable } ]
    [ [ , ] FORMAT | NOFORMAT ]
    [ [ , ] { INIT | NOINIT } ]
    [ [ , ] MEDIADESCRIPTION = { 'text' | @text_variable } ]
    [ [ , ] MEDIANAME = { media_name | @media_name_variable } ]
    [ [ , ] MEDIAPASSWORD = { mediapassword | @mediapassword_variable } ]
    [ [ , ] NAME = { backup_set_name | @backup_set_name_var } ]
    [ [ , ] { NOSKIP | SKIP } ]
    [ [ , ] { NOREWIND | REWIND } ]
    [ [ , ] { NOUNLOAD | UNLOAD } ]
    [ [ , ] RESTART ]
    [ [ , ] STATS [ = percentage ] ]
]
```

参数说明

❶ DATABASE：指定一个完整的数据库备份。假如指定了一个文件或文件组的列表，那么仅有这些被指定的文件或文件组被备份。

❷ { database_name | @database_name_var }：指定一个数据库，在该数据库中对事务日志、部分数据库或完

整的数据库进行备份。如果作为变量（@database_name_var）提供，则可将该名称指定为字符串常量（@database_name_var = database name）或字符串数据类型（ntext 或 text 数据类型除外）的变量。

❸ < backup_device >：指定备份操作时要使用的逻辑或物理备份设备。可以是下列一种或多种形式。

- { logical_backup_device_name } | { @logical_backup_device_name_var }：由 sp_addumpdevice 创建的备份设备的逻辑名称（数据库将备份到该设备中），其名称必须遵守标识符规则。如果将其作为变量（@logical_backup_device_name_var）提供，则可将该备份设备名称指定为字符串常量 （@logical_backup_device_name_var = logical backup device name）或字符串数据类型（ntext 或 text 数据类型除外）的变量。
- { DISK | TAPE } ='physical_backup_device_name' | @physical_backup_device_name_var：允许在指定的磁盘或磁带设备上创建备份。在执行 BACKUP 语句之前不必存在指定的物理设备。如果存在物理设备且 BACKUP 语句中没有指定 INIT 选项，则备份将追加到该设备。

注意： 当指定 TO DISK 或 TO TAPE 时，要输入完整路径和文件名。例如，DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL\BACKUP\backup.dat'。

❹ n：表示可以指定多个备份设备的占位符。备份设备数目的上限为 64。

❺ BLOCKSIZE = {blocksize | @blocksize_variable}：用字节数来指定物理块的大小。在 Windows NT 系统中，默认设置是设备的默认块大小。一般情况下，当 SQL Server 选择适合于设备的块大小时不需要此参数。在基于 Windows 2000 的计算机上，默认设置是 65536（64KB，是 SQL Server 支持的最大大小）。

❻ DESCRIPTION = { 'text' | @text_variable }：指定描述备份集的自由格式文本。该字符串最长可以有 255 个字符。

❼ DIFFERENTIAL：指定数据库备份或文件备份应该与上一次完整备份后改变的数据库或文件部分保持一致。差异备份一般会比完整备份占用更少的空间。对于上一次完整备份时备份的全部单个日志，使用该选项可以不必再进行备份。

❽ EXPIREDATE = { date | @date_var }：指定备份集到期和允许被重写的日期。如果将该日期作为变量（@date_var）提供，则可以将该日期指定为字符串常量（@date_var = date）、字符串数据类型变量（ntext 或 text 数据类型除外）、smalldatetime 或者 datetime 变量，并且该日期必须符合已配置的系统 datetime 格式。

❾ RETAINDAYS = {days | @days_var }：指定必须经过多少天才可以重写该备份媒体集。假如用变量（@days_var）指定，该变量必须为整型。

❿ PASSWORD = { password | @password_variable }：为备份集设置密码。PASSWORD 是一个字符串。如果为备份集定义了密码，则必须提供该密码才能对该备份集执行任何还原操作。

⓫ FORMAT：指定应将媒体头写入用于此备份操作的所有卷。任何现有的媒体头都被重写。FORMAT 选项使整个媒体内容无效，并且忽略任何现有的内容。

⓬ NOFORMAT：指定媒体头不应写入所有用于该备份操作的卷中，并且不要重写该备份设备，除非指定了 INIT。

⓭ INIT：指定应重写所有备份集，但是保留媒体头。如果指定了 INIT，将重写那个设备上所有现有的备份集数据。

当遇到以下几种情况之一时不重写备份媒体：

- 媒体上的备份设置没有全部过期。
- 如果 BACKUP 语句给出了备份集名，该备份集名与备份媒体上的名称不匹配。

⓮ NOINIT：表示备份集将追加到指定的磁盘或磁带设备上，以保留现有的备份集。NOINIT 是默认设置。

⓯ MEDIADESCRIPTION = { 'text' | @text_variable }：指定媒体集的自由格式文本描述，最多为 255 个字符。

⓰ MEDIANAME={media_name|@media_name_variable}：为整个备份媒体集指定媒体名，最多为 128 个字符。假如指定了 MEDIANAME，则它必须与以前指定的媒体名相匹配，该媒体名已存在于备份卷中。假如没有

指定 MEDIANAME，或指定了 SKIP 选项，将不会对媒体名进行验证检查。

⑰ MEDIAPASSWORD = {mediapassword | @mediapassword_variable}：为媒体集设置密码。MEDIAPASSWORD 是一个字符串。

⑱ NAME = { backup_set_name | @backup_set_name_var }：指定备份集的名称，最长可达 128 个字符。假如没有指定 NAME，它将为空。

⑲ NOSKIP：指定 BACKUP 语句在可以重写媒体上的所有备份集之前先检查其过期日期。

⑳ SKIP：禁用备份集过期和名称检查，这些检查一般由 BACKUP 语句执行以防重写备份集。

㉑ NOREWIND：指定 SQL Server 在备份操作完成后使磁带保持打开。

㉒ REWIND：指定 SQL Server 将释放磁带。如果 NOREWIND 和 REWIND 均未指定，则默认设置为 REWIND。

㉓ NOUNLOAD：指定不在备份后从磁带驱动器中自动卸载磁带。设置始终为 NOUNLOAD，直到指定 UNLOAD 为止。该选项只用于磁带设备。

㉔ UNLOAD：指定在备份完成后自动倒带并卸载磁带。启动新用户会话时其默认设置为 UNLOAD。该设置一直保持到用户指定了 NOUNLOAD 时为止。该选项只用于磁带设备。

㉕ RESTART：指定 SQL Server 重新启动一个被中断的备份操作。因为 RESTART 选项在备份操作被中断处重新启动该操作，所以节省了时间。若要重新启动一个特定的备份操作，可重复整个 BACKUP 语句并且加入 RESTART 选项。不一定非要使用 RESTART 选项，但是它可以节省时间。

㉖ STATS [= percentage]：每当另一个 percentage 结束时显示一条消息，通常用于测量进度。如果省略 percentage，SQL Server 将每完成 10 个百分点显示一条消息。

（1）在项目中创建 UserDao 类，定义数据库连接、更新和关闭的方法。

（2）创建 index.jsp 页面，读取 SQL Server master 数据库中 sysdatabases 数据表中的数据；创建 form 表单，添加下拉列表，将 sysdatabases 数据表中的数据添加到下拉列表中，添加文本框用于设置备份文件的存储位置和路径。关键代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java" import="java.sql.*" errorPage="" %>
    <jsp:useBean id="dao" scope="request" class="com.pkh.dao.UserDao"/>
<%
        ResultSet rs = dao.selectStatic("select name from dbo.sysdatabases");        //查询数据表中的数据
        String path = request.getParameter("path");                                  //获取存储的路径和名称
        String dbase=request.getParameter("select");                                 //获取要备份数据库名称
%>
<form name="form1" method="post" action="">
        <select name="select">
        <%
                while (rs.next()) {                                                  //循环输出查询结果集中的数据
                        String database = rs.getString(1);                           //获取数据表名称
                        out.println("<option id='" + database + "'>" + database+ "</option>");
                }
                if (path != null&&dbase!=null) {
                        dao.executeUpdate("backup database "+dbase+" to disk='"+path+"'");//执行备份操作
                        out.print("<script language='javascript'>alert('备份完成');</script>");
                }
                dao.closeConnection();                                               //关闭数据库
        %>
        </select>
        <input type="text" name="path">
        <input type="submit" name="Submit" value="备份">
</form>
```

秘笈心法

心法领悟 072：SQL Server 数据库备份的 4 种类型。

SQL Server 数据库备份分为 4 种类型，分别为完全备份、事务日志备份、差异备份、文件和文件组备份。完全备份是将整个数据库（包括表、索引、视图等所有数据库对象）都进行备份；事务日志备份只需要复制自上次备份以来对数据库所做的改变；差异备份只包含自完全备份以来所改变的数据库；文件和文件组备份只备份数据库中的一部分。

实例 073　SQL Server 数据库的恢复

光盘位置：光盘\MR\03\073

中级　实用指数：★★★

实例说明

应用数据恢复技术可以将被破坏的数据库重新还原，以保证系统的安全。本实例实现的是将备份的数据库重新还原，运行结果如图 3.16 所示。

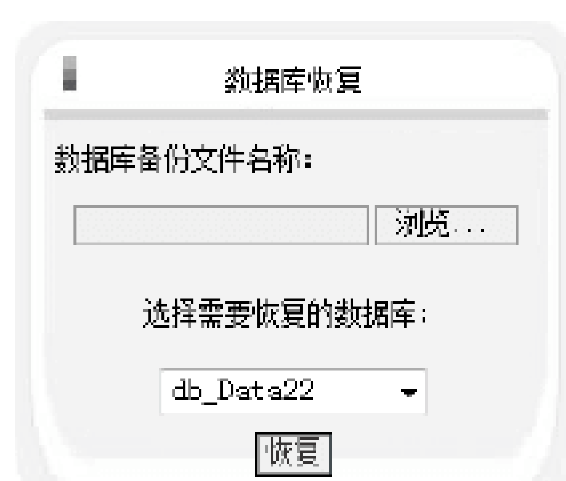


图 3.16　数据库恢复

注意： 恢复数据库备份将重新创建数据库和备份完成时数据库中存在的所有相关文件。如果要恢复创建数据库备份后所产生的事务，必须使用事务日志备份或差异备份。

本实例运用 SQLDMO.Restore 对象恢复使用 BACKUP 命令所做的整个数据库备份。

RESTORE 的语法如下。

```
RESTORE DATABASE { database_name | @database_name_var }
[ FROM < backup_device > [ ,...n ] ]
[ WITH
      [ RESTRICTED_USER ]
      [ [ , ] FILE = { file_number | @file_number } ]
      [ [ , ] PASSWORD = { password | @password_variable } ]
      [ [ , ] MEDIANAME = { media_name | @media_name_variable } ]
      [ [ , ] MEDIAPASSWORD = { mediapassword | @mediapassword_variable } ]
      [ [ , ] MOVE 'logical_file_name' TO 'operating_system_file_name' ]
         [ ,...n ]
      [ [ , ] KEEP_REPLICATION ]
      [ [ , ] { NORECOVERY | RECOVERY | STANDBY = undo_file_name } ]
      [ [ , ] { NOREWIND | REWIND } ]
      [ [ , ] { NOUNLOAD | UNLOAD } ]
      [ [ , ] REPLACE ]
      [ [ , ] RESTART ]
      [ [ , ] STATS [ = percentage ] ]
]
```

参数说明

❶ DATABASE：指定备份还原整个数据库。如果指定了文件和文件组列表，则只还原那些文件和文件组。

❷ {database_name | @database_name_var}：将日志或整个数据库还原到的数据库。如果将其作为变量（@database_name_var）提供，则可将该名称指定为字符串常量（@database_name_var = database name）或字符

串数据类型（ntext 或 text 数据类型除外）的变量。

❸ FROM：指定从中还原备份的备份设备。如果没有指定 FROM 子句，则不会发生备份还原，而是恢复数据库。可用省略 FROM 子句的方法尝试恢复通过 NORECOVERY 选项还原的数据库，或切换到一台备用服务器上。如果省略 FROM 子句，则必须指定 NORECOVERY、RECOVERY 或 STANDBY。

❹ < backup_device >：指定还原操作要使用的逻辑或物理备份设备。可以是下列一种或多种形式。

- {'logical_backup_device_name' | @logical_backup_device_name_var}：由 sp_ addumpdevice 创建的备份设备（数据库将从该备份设备还原）的逻辑名称，该名称必须符合标识符规则。如果作为变量（@logical_ backup_device_name_var）提供，则可以指定字符串常量（@logical_backup_device_name_var = logical_backup_ device_ name）或字符串数据类型（ntext 或 text 数据类型除外）的变量作为备份设备名。
- {DISK | TAPE } = 'physical_backup_device_name' | @physical_backup_device_name_ var：允许从命名磁盘或磁带设备还原备份。磁盘或磁带的设备类型应该用设备的真实名称（例如，完整的路径和文件名）来指定：DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL\BACKUP\Mybackup.dat'或 TAPE = '\\.\TAPE0'。如果指定为变量（@physical_backup_device_name_var），则设备名称可以是字符串常量（@physical_ backup_ device_name_var = 'physical_backup_device_name'）或字符串数据类型（ntext 或 text 数据类型除外）的变量。

❺ n：表示可以指定多个备份设备和逻辑备份设备的占位符。备份设备或逻辑备份设备最多可以为 64 个。

❻ RESTRICTED_USER：限制只有 db_owner、dbcreator 或 sysadmin 角色的成员才能访问新近还原的数据库。

❼ FILE = {file_number | @file_number}：标识要还原的备份集。例如，file_number 为 1 表示备份媒体上的第 1 个备份集，file_number 为 2 表示第 2 个备份集。

❽ PASSWORD = { password | @password_variable }：提供备份集的密码。PASSWORD 是一个字符串。如果在创建备份集时提供了密码，则从备份集执行还原操作时必须提供密码。

❾ MEDIANAME = {media_name | @media_name_variable}：指定媒体名称。如果提供媒体名称，该名称必须与备份卷上的媒体名称相匹配，否则还原操作将终止。如果 RESTORE 语句没有给出媒体名称，将不对备份卷执行媒体名称匹配检查。

❿ MEDIAPASSWORD = {mediapassword | @mediapassword_variable}：提供媒体集的密码。MEDIAPASSWORD 是一个字符串。如果格式化媒体集时提供了密码，则访问该媒体集上的任何备份集时都必须提供该密码。

⓫ MOVE 'logical_file_name' TO 'operating_system_file_name'：指定应将给定的 logical_file_name 移到 operating_system_file_name。默认情况下，logical_file_name 将还原到其原始位置。如果使用 RESTORE 语句将数据库复制到相同或不同的服务器上，则可能需要使用 MOVE 选项重新定位数据库文件以避免与现有文件冲突。可以在不同的 MOVE 语句中指定数据库内的每个逻辑文件。

⓬ n：占位符，表示可通过指定多个 MOVE 语句移动多个逻辑文件。

⓭ KEEP_REPLICATION：指示还原操作在将发布的数据库还原到创建它的服务器以外的服务器上时保留复制设置。当设置复制与日志传送一同使用时，需使用 KEEP_REPLICATION。这样，当在备用服务器上还原数据库或日志备份并且恢复数据库时，可防止删除复制设置。还原备份时若指定了该选项，则不能选择 NORECOVERY 选项。

⓮ NORECOVERY：指示还原操作不回滚任何未提交的事务。如果需要应用另一个事务日志，则必须指定 NORECOVERY 或 STANDBY 选项。如果 NORECOVERY、RECOVERY 和 STANDBY 均未指定，则默认为 RECOVERY。当还原数据库备份和多个事务日志，或需要多个 RESTORE 语句时（例如在完整数据库备份后进行差异数据库备份），SQL Server 要求在除最后的 RESTORE 语句外的其他所有语句中使用 WITH NORECOVERY 选项。

⓯ RECOVERY：指示还原操作回滚任何未提交的事务。在恢复进程后即可随时使用数据库。如果安排了后续 RESTORE 操作（RESTORE LOG 或从差异数据库备份 RESTORE DATABASE），则应改为指定 NORECOVERY

或 STANDBY。

⑯ STANDBY = undo_file_name：指示撤销文件名以便可以取消恢复效果。撤销文件的大小取决于因未提交的事务所导致的撤销操作量。如果 NORECOVERY、RECOVERY 和 STANDBY 均未指定，则默认为 RECOVERY。STANDBY 允许将数据库设定为在事务日志还原期间只能读取，并且可用于备用服务器，或用于需要在日志还原操作之间检查数据库的特殊恢复情形。

⑰ NOREWIND：指定 SQL Server 在备份操作完成后使磁带保持打开状态，以防止其他过程访问磁带。直到执行 REWIND 或 UNLOAD 语句，或服务器关闭时，才释放该磁带。通过查询 master 数据库中的 sysopentapes 表可查找当前打开的一系列磁带。NOREWIND 与 NOUNLOAD 关键字相同。该选项只用于磁带设备，如果对 RESTORE 使用非磁带设备，将忽略该选项。

⑱ REWIND：指定 SQL Server 将释放磁带和倒带。如果 NOREWIND 和 REWIND 均未指定，则默认设置为 REWIND。该选项只用于磁带设备，如果对 RESTORE 使用非磁带设备，将忽略该选项。

⑲ NOUNLOAD：指定不在 RESTORE 后从磁带机中自动卸载磁带。设置始终为 NOUNLOAD，直到指定 UNLOAD 为止。该选项只用于磁带设备，如果对 RESTORE 使用非磁带设备，将忽略该选项。

⑳ UNLOAD：指定在还原完成后自动倒带并卸载磁带。启动新用户会话时其默认设置为 UNLOAD。设置始终为 UNLOAD，直到指定 NOUNLOAD 为止。该选项只用于磁带设备，如果对 RESTORE 使用非磁带设备，将忽略该选项。

㉑ REPLACE：指定即使存在另一个具有相同名称的数据库，SQL Server 也应该创建指定的数据库及其相关文件。在这种情况下将删除现有的数据库。如果没有指定 REPLACE 选项，则将进行安全检查以防止意外重写其他数据库。

㉒ RESTART：指定 SQL Server 应重新启动被中断的还原操作。RESTART 从中断点重新启动还原操作。

㉓ STATS [= percentage]：每当另一个 percentage 结束时显示一条消息，通常并用于测量进度。如果省略 percentage，则 SQL Server 每完成 10 个百分比显示一条消息。

（1）创建 UserDao 类，定义数据库连接、更新和关闭的方法。

（2）创建 index.jsp 页面，读取 SQL Server master 数据库中 sysdatabases 数据表中的数据；创建 form 表单，添加下拉列表，将 sysdatabases 数据表中的数据添加到下拉列表中，添加文本框用于设置备份文件的存储位置和路径。关键代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java" import="java.sql.*" errorPage="" %>
    <jsp:useBean id="dao" scope="request" class="com.pkh.dao.UserDao"/>
<%
        ResultSet rs = dao.selectStatic("select name from dbo.sysdatabases");        //执行查询语句
        String path = request.getParameter("path");                                   //获取指定备份文件的位置
        String dbase = request.getParameter("select");                                //获取要恢复的指定文件
%>
<form name="form1" method="post" action="">
        <input type="text" name="path">
        <select name="select">
        <%
                while (rs.next()) {                                                   //循环输出结果集
                        String database = rs.getString(1);                            //输出数据表名称
                        out.println("<option id='" + database + "'>" + database+ "</option>");
                }
                if (path != null && dbase != null) {                                  //判断路径和数据表是否存在
                        dao.executeUpdate("restore database " + dbase + " from disk='" + path + "'");   //执行恢复操作
                        out.print("<script language='javascript'>alert('恢复完成');</script>");
                }
                dao.closeConnection();                                                //关闭数据库
        %>
        </select>
        <input type="submit" name="Submit" value="恢复">
```

秘笈心法

心法领悟 073：SQLServer 数据库恢复支持的 4 种类型。

与数据库备份对应，数据库恢复也支持 4 种类型，分别是还原整个数据库的完整数据库恢复；差异数据库恢复；事务日志还原；个别文件和文件组恢复。

实例 074　MySQL 数据库的备份

光盘位置：光盘\MR\03\074

中级

实用指数：★★☆

实例说明

作为一种十分流行的数据库，MySQL 被广泛应用在 Web 开发中，因此掌握其备份技术非常重要。备份 MySQL 数据库的方法很多，可以通过 MySQL Tools 或者 phpMy Admin 管理工具进行备份，也可以通过命令进行备份。本实例将使用 MySQL DUMP 命令备份数据库，运行结果如图 3.17 所示。

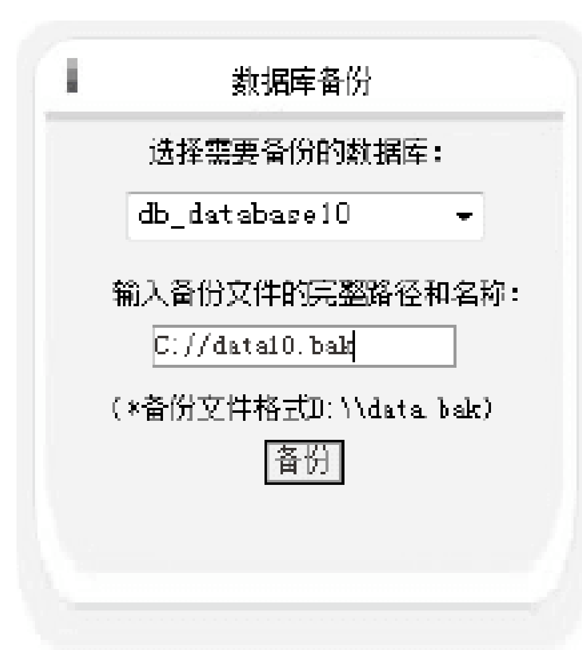


图 3.17　MySQL 数据库备份

本实例主要用到 java.lang 包中的 Runtime、Process 和 StringBuffer 类。首先通过 getRuntime()方法获取与当前 Java 应用程序相关的 Runtime 对象；然后应用 exec()方法，执行 MYSQLDUMP 命令；接着应用 Process 类中的 getInputStream()方法获取子进程的输入流；最后应用 StringBuffer 类中的 append()方法将流中的数据追加到指定的字符序列中，完成数据库的备份。下面对使用的方法进行详细的讲解。

（1）getRuntime()方法

语法如下：

```
public static Runtime getRuntime()
```

返回与当前 Java 应用程序相关的 Runtime 对象。Runtime 类的大多数方法都是实例方法，并且必须根据当前的运行对象对其进行调用。

（2）exec()方法

语法如下：

```
public Process exec(String command)
                    throws IOException
```

在单独的进程中执行指定的字符串命令。该方法将返回一个新的 Process 对象，用于管理子进程。

参数说明

command：一条指定的系统命令。

以下情况会抛出异常。

- SecurityException：如果安全管理器存在，并且其 checkExec()方法不允许创建子进程。
- IOException：如果发生 I/O 错误。
- NullPointerException：如果 command 为 null。

（3）getInputStream()方法

语法如下：

```
public abstract InputStream getInputStream()
```

获取子进程的输入流。输入流获得由该 Process 对象表示的进程的标准输出流。返回连接到子进程正常输出的输入流。

（4）append()方法

语法如下：

```
public StringBuffer append(String str)
```

将指定的字符串追加到调用该方法的字符串序列。按顺序追加 String 变量中的字符，使此序列增加该变量的长度。如果 str 为 null，则追加 4 个字符"null"。返回值为此对象的一个引用。

参数说明

str：一个指定的字符串。

如果此字符序列的长度在执行 append()方法前为 n，且 k 小于 n，则新字符序列中索引 k 处的字符等于原序列中索引 k 处的字符；否则等于参数 str 中索引 k−n 处的字符。

MYSQLDUMP 命令的语法如下。

```
mysqldump –uUser –pPass DataBase > Path
```

其中，User 是用户名，Pass 是密码，DataBase 是数据库名，Path 是数据库备份存储的位置。

注意： 要通过 MYSQLDUMP 命令来备份 MySQL 数据库，必须对计算机的环境变量进行设置。在“我的电脑”上右击，在弹出的快捷菜单中选择“属性”命令，弹出“系统属性”对话框。选择“高级”选项卡，单击“环境变量”按钮，弹出“环境变量”对话框。在“用户变量”列表框中找到变量 PATH，单击“编辑”按钮，弹出“编辑用户变量”对话框。在“变量值”文本框中输入 D:\Program Files\MySQL\MySQL Server 5.0\bin（MySQL 数据库中 bin 文件夹的安装路径，根据自己安装 MySQL 数据库的位置而定），然后单击“确定”按钮即可。

■ 设计过程

（1）创建 UserDao 类，定义数据库连接、更新和关闭的方法，并且在该类中定义数据库备份的方法 mysqldump()。本实例中连接的是 MySQL 的系统数据库 information_schema。

（2）创建 index.jsp 页面，读取 information_schema 数据库中 SCHEMATA 数据表中的数据；创建 form 表单，添加下拉列表，将 SCHEMATA 数据表中的数据添加到下拉列表中，添加文本框用于设置备份文件的存储位置和路径。关键代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java" import="java.sql.*" errorPage="" %>
    <jsp:useBean id="dao" scope="request" class="com.pkh.dao.UserDao"/>
<%
        ResultSet rs = dao.selectStatic("SELECT schema_name FROM SCHEMATA");     //执行查询操作，获取结果集
        String path = request.getParameter("path");                              //获取备份文件存储的位置
        String dbase = request.getParameter("select");                           //获取要备份的数据库
        if("备份".equals(request.getParameter("Submit"))){
                if (path != null && dbase != null) {
                        if (dao.mysqldump(dbase,path)) {                         //调用方法，执行数据库的备份
                                out.print("<script language='javascript'>alert('备份完成');</script>");
                        }
                }
        }
%>
<form name="form1" method="post" action="">
        <select name="select">
        <%
                while (rs.next()) {
                        String database = rs.getString(1);
                        out.println("<option id='" + database + "'>" + database + "</option>");
                }
                dao.closeConnection();
        %>
        </select>
        <input type="text" name="path">
        <input type="submit" name="Submit" value="备份">
</form>
```

■

心法领悟 074：使用正确的文件地址。

很多程序都会涉及文件地址的问题，例如 D:\\Data\\db_database04.txt 这样的文件路径，需要注意的是在文

件地址中要使用双斜杠“\\”，而不是单斜杠“\”。



实例 075　MySQL 数据库的恢复

光盘位置：光盘\MR\03\075

中级

实用指数：★★☆

实例说明

对于大型 Web 项目来说，具有良好的数据备份与恢复功能是非常重要的，一旦数据遭到破坏，即可将备份的数据恢复到数据库系统中，以此保证系统的正常运行，以免数据丢失。本实例实现的是将实例 074 备份的 MySQL 数据库文件恢复到数据库中，运行结果如图 3.18 所示。

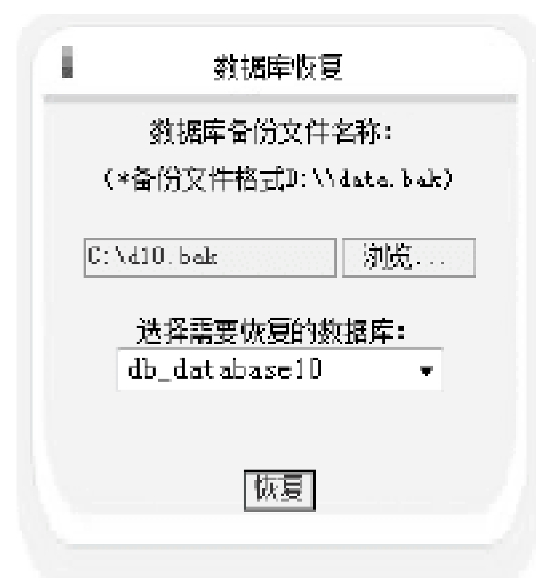


图 3.18　数据库恢复

关键技术

本实例实现数据库的恢复使用的是 MySQL 数据库中的恢复数据命令，其语法如下。

```
mysql –uUser –pPass DataBase <Path
```

其中，User 指定数据库用户名；Pass 指定密码；DataBase 指定备份的数据库；Path 是备份文件存储的位置。

设计过程

（1）创建 UserDao 类，定义数据库连接、更新和关闭的方法，并且在该类中定义数据库恢复的方法 mysql()。本实例中连接的是 MySQL 的系统数据库 information_schema。

（2）创建 index.jsp 页面，读取 information_schema 数据库中 SCHEMATA 数据表中的数据；创建 form 表单，添加下拉列表，将 SCHEMATA 数据表中的数据添加到下拉列表中；添加文本框，用于设置备份文件的存储位置和路径。其关键代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java" import="java.sql.*" errorPage="" %>
    <jsp:useBean id="dao" scope="request" class="com.pkh.dao.UserDao"/>
<%
        ResultSet rs = dao.selectStatic("SELECT schema_name FROM SCHEMATA");          //执行查询语句
        String path = request.getParameter("path");                                   //获取恢复文件存储路径
        String dbase = request.getParameter("select");                                //获取要恢复的数据库
        if ("恢复".equals(request.getParameter("Submit"))) {
                if (path != null && dbase != null) {
                        if (dao.mysqlresume(dbase,path)) {                             //执行恢复操作
                                out.print("<script language='javascript'>alert('恢复完成');</script>");
                        }
                }
        }
%>
<form name="form1" method="post" action="">
        <input name="path" type="file" size="18">
        <select name="select">
        <%
                while (rs.next()) {
                        String database = rs.getString(1);
                        out.println("<option id='" + database + "'>" + database + "</option>");
                }
                dao.closeConnection();
        %>
        </select>
        <input type="submit" name="Submit" value="恢复">
</form>
```

心法领悟 075：mysqldump 工具。

mysqldump 是 MySQL 自带的一款非常方便的小工具，存放在 MySQL 安装目录的 bin 下。不是 MySQL 命令不能在 MySQL 中执行，在命令标识符中输入 mysqldump - help 命令可以测试 mysqldump 所支持的选项。

3.3　数据库的添加、删除与更新操作

实例 076　将员工信息添加到员工表

光盘位置：光盘\MR\03\076

中级　实用指数：★★☆

实例说明

对数据的插入操作是每个程序必不可少的。本实例实现的是向数据库中插入单条记录。在数据插入的过程中，为了保证程序的完整，可以通过 JavaScript 脚本进行验证。当用户单击“提交”按钮时，会将用户添加的信息写入到数据库中。本实例运行结果如图 3.19 所示。

图 3.19　将员工信息添加到员工表

关键技术

本实例在实现数据添加时，使用的是 INSERT 插入语句。INSERT 语句的语法如下。

```
INSERT INTO  表名[(字段名 1,字段名 2,…)]
VALUES(属性值 1,属性值 2, …)
```

例如，向数据表 tb_emp（包含字段 id、name、sex、department）中插入数据，代码如下：

```
insert into tb_emp values(2,'lili','女','销售部');
```

（1）在项目中创建 Emp 类，该类中包含与用户表 tb_emp 对应的属性，并包含属性的 setXXX()与 getXXX()方法。

（2）在项目中创建 EmpDao 类，在该类中定义向员工表插入数据的方法 insertEmp()，该方法以 Emp 对象作为参数。具体代码如下：

```
public String insertEmp(Emp emp){
        conn = getConnection();
        try {
                PreparedStatement statement = conn.prepareStatement("insert into tb_emp values(null,?,?,?,?,?,?)");     //定义插入员工信息的 SQL 语句
                statement.setString(1, emp.getName());                //设置预处理语句参数值
                statement.setInt(2, emp.getAge());                    //设置预处理语句参数值
                statement.setString(3, emp.getSex());
                statement.setString(4, emp.getDept());
                statement.setString(5, emp.getPhone());
                statement.setString(6, emp.getRemark());
                statement.executeUpdate();                            //执行插入语句
                conn.close();                                         //关闭数据库连接
```

```
            return "数据插入成功";                              //返回执行结果
        } catch (Exception e) {
            e.printStackTrace();
            return "数据插入失败";
        }
}
```

秘笈心法

心法领悟 076：灵活地使用预处理语句。

本实例在执行插入数据时，使用了预处理语句。与普通 SQL 语句相比，预处理语句的第一个优点是安全，第二个优点是可以提高访问数据库的速度。因此，提倡使用预处理语句来操作数据库。

实例 077　在添加数据时进行数据验证

光盘位置：光盘\MR\03\077

中级

实用指数：★★★☆

实例说明

很多网站都要求用户在注册网站时保证用户名的唯一性，为了达到这样的目的，会提供类似用户名验证的功能。本实例实现的是在添加员工信息时进行用户名的验证，来保证用户名的唯一性，运行结果如图 3.20 所示。

图 3.20　在添加数据时进行数据验证

关键技术

本实例实现的是用户名验证，当用户单击“验证”按钮时，会从数据库中查询数据，如果数据库中包含相关信息，会给出提示信息。

设计过程

（1）在项目中创建类 Emp，该类中的属性与 tb_emp 表中的字段一一对应，并包含属性的 setXXX()与 getXXX()方法。具体代码参见配书光盘中的源程序。

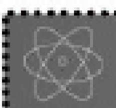
（2）在项目中创建类 EmpDao，在该类中定义按照用户名查询用户的方法 findName()。该方法包含一个 String 类型的参数，用于指定要查询的用户名。具体代码如下：

```
public int findName(String name){
        conn = getConnection();                                     //获取数据库连接
        int id = 0;
        try {
            Statement statement = conn.createStatement();           //获取 Statement 对象
            String sql = "select id from tb_emp where name ='"+name+"'";   //定义查询 SQL 语句
          ResultSet rest = statement.executeQuery(sql);
            while(rest.next()){                                     //循环遍历查询结果集
                id = rest.getInt(1);                                //获取查询结果
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return id;                                                  //返回查询结果
}
```

心法领悟 077：实现 Eclipse 中的代码格式化。

在编写代码时，可能会没有注意到格式化的问题，从而使代码显得比较凌乱。如果使用 Eclipse 开发工具，可以通过快捷键 Ctrl+Shift+F 来对 Eclipse 中的代码进行格式化。



实例 078 插入用户登录日志信息

光盘位置：光盘\MR\03\078

中级

实用指数：★★☆

实例说明

几乎所有的网站都包含有登录模块。本实例实现的是在验证用户是否为合法用户的同时，将用户登录的时间插入到数据库中，以此可以统计网站的访问流量。本实例的运行结果如图 3.21 所示。



图 3.21　插入用户登录日志信息

关键技术

要实现将用户登录的时间写入到数据库中，并对日期进行格式化，可以使用 SimpleDateFormat 类的 format()方法完成。

SimpleDateFormat 类是一个与语言环境有关的用来格式化和解析日期的具体类，其常用构造方法如下。

- 用默认的模式和默认语言环境的日期格式符号构造 SimpleDateFormat 类。语法如下：

```
SimpleDateFormat()
```

- 用给定的模式和默认语言环境的日期格式符号构造 SimpleDateFormat 类。语法如下：

```
SimpleDateFormat(String pattern)
```

参数说明

pattern：描述日期和时间格式的模式。

该类的 format()方法可将给定的 Date 格式化为日期/时间字符串，并以 String 类型返回格式化后的字符串。该方法的语法如下：

```
format(Date date)
```

参数说明

date：要格式化为时间字符串的时间值。

（1）在项目中创建类 UserDao，在该类中定义按照用户名与密码查询用户的方法 findUser()。该方法包含两个 String 类型的参数，分别用于指定用户名与密码。具体代码如下：

```
public int findUser(String userName,String passWord){
        int id = 0;
        conn = getConnection();                                          //获取数据库连接
        try {
                Statement statement = conn.createStatement();            //获取 Statement 对象
                String sql = "select id from tb_user where userName ='"+userName+"' and passWord ='"+passWord+"'";  //根据用户名与密码查询数据库
                ResultSet rest = statement.executeQuery(sql);            //执行查询 SQL 语句
                while(rest.next()){                                      //循环遍历查询结果集
                        id = rest.getInt(1);                             //获取查询结果
                }
        } catch (Exception e) {
                e.printStackTrace();
        }
        return id;                                                       //返回查询结果
}
```

（2）在 UserDao 类中定义获取当前系统时间的方法 getDateTime()，该方法将格式化后的系统时间返回。具体代码如下：

```
public String getDateTime(){
        SimpleDateFormat format;                                         //定义 SimpleDateFormat 对象
        Date date = null;
        Calendar myDate = Calendar.getInstance();                        //创建 Calendar 对象
        myDate.setTime(new java.util.Date());                            //设置对象时间
```

```
        date = myDate.getTime();
        format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");                    //定义日期格式
        String strRtn = format.format(date);                                      //将日期进行格式化
        return strRtn;                                                            //返回格式化结果
}
```

（3）在该类中定义 insertUserInfo()方法，用于向用户日志表中插入数据。具体代码如下：

```
public void insertUserInfo(UserInfo info){
        conn = getConnection();                                                   //获取数据库连接
        try {
                PreparedStatement pstatement = conn.prepareStatement("insert into tb_userinfo values(null,?,?)");   //定义向用户表中插入数据的 SQL 语句
                pstatement.setString(1, info.getUserName());                      //设置预处理语句参数值
                pstatement.setString(2, getDateTime());
                pstatement.executeUpdate();                                       //执行插入操作
        } catch (SQLException e) {
                e.printStackTrace();
        }
}
```

秘笈心法

心法领悟 078：定义日期格式时注意大小写。

本实例将当前系统时间以“年、月、日 时:分:秒”的格式定义。使用 SimpleDateFormat 类设置日期格式时，一定要注意大小写。例如，“MM”表示月份，“mm”表示分钟，如果没有注意区分大小写，就会得不到想要的结果。

实例 079　生成有规律的编号

光盘位置：光盘\MR\03\079

中级

实用指数：★★☆

实例说明

数据库中的每个表都要有一个主键，虽然在数据库中可设置主键的自动增长功能，但是有些程序需要生成有一定规律的主键，例如 CS201012220001 等，这样的主键通过数据库的自动增长功能是无法实现的，需要通过程序来实现。本实例将实现生成有规律的商品编号，自动生成的商品编号由 CS 开头，之后是商品插入的时间，接下来是商品的排序号。本实例的运行结果如图 3.22 所示。



图 3.22　有规律编号

商品编号由字母 CS、系统日期和 5 位数字组成。首先判断采购信息表中的采购编号是否为空，如果为空，则采购编号等于字母 CS+系统日期+00001；如果不为空，则先查找数据表中最大的采购编号，此时采购编号等于字母 CS+系统日期+5 位数字编码。在实现编码加 1 时，数字前面的 0 将被忽略。实现数字前加 0 的格式，可以采用字符串的 format()方法。

format()方法用于输出指定格式的字符串，其语法如下。

```
String.format("%05d", ID + 1)
```

参数说明

❶ "%05d"：表示格式化字符，设定数字编码的位数为 5 位，如果不足 5 位前面补 0。

❷ ID：数字格式的变量，将其加 1 实现编号增值。

设计过程

（1）在项目中创建类 WareDao，在该类中定义从商品表中查询所有商品信息的方法 getWares()，该方法以 List 集合为返回值。

（2）在 WareDao 类中定义向商品表 tb_ware 中插入数据的方法 insertWare()，该方法包含一个与商品表对应的 JavaBean 对象并以其为参数。具体代码如下：

```
public void insertWare(Ware ware){
	conn = getConnection();                                                              //获取数据库连接
	try {
		PreparedStatement statement = conn.prepareStatement("insert into tb_ware values(?,?,?,?,?,?)");  //定义插入数据库 SQL 语句
		statement.setString(1, ware.getMid());                                      //设置预处理语句参数
		statement.setString(2, ware.getSpec());
		statement.setString(3, ware.getCasing());
		statement.setString(4, ware.getUnit());
		statement.setInt(5,ware.getCount());
		statement.setString(6, ware.getName());
		statement.executeUpdate();                                                  //执行预处理语句实现插入数据操作
	} catch (Exception e) {

		e.printStackTrace();
}
}
```

（3）当用户单击页面中的"提交"按钮后，系统会提交至名为 WareServlet 的 Servlet 处理请求。WareServlet 中的关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
		throws ServletException, IOException {
	WareDao dao = new WareDao();
	String sDate = dao.getDateTime();                                   //调用获取系统时间方法
	List list = dao.getWares();                                         //获取商品表中全部的商品
	int ID = 0;
	String sid = "";
	for (int i = 0; i < list.size(); i++) {                             //循环遍历查询结果集
		Ware ware = (Ware) list.get(i);                                 //获取商品
		sid = ware.getMid();                                            //获取商品编号
	}
	if (list.size() == 0) {                                             //如果商品集合为空
		sid = "CS" + sDate.replace("-", "") + "00001";                  //定义商品编号
	} else {                                                            //如果商品集合不为空
		sid = sid.trim();
		ID = Integer.parseInt(sid.substring(sid.length() - 5));         //截取商品编号中的后 5 位
		sid = sid.substring(0, sid.length() - 5)
				+ String.format("%05d", ID + 1);                        //商品编号
	}
	Ware ware = new Ware();                                             //定义与商品表对应的 JavaBean 对象
	ware.setMid(sid);
	ware.setName(request.getParameter("nameTextfield"));                //设置 JavaBean 属性
	ware.setCasing(request.getParameter("casingTextfield"));
	ware.setSpec(request.getParameter("specTextfield"));
	ware.setUnit(request.getParameter("unitTextfield"));
	ware.setCount(Integer.parseInt(request.getParameter("countTextfield")));
	dao.insertWare(ware);                                               //调用添加数据方法
	request.setAttribute("message", "数据添加成功");                     //向 request 对象中添加信息
	request.getRequestDispatcher("index.jsp").forward(request, response);  //设置页面转发地址
}
```

心法领悟 079：不要忽略负数。

在程序开发中经常使用的都是正数，负数因为使用得少而常常被忽略。例如，"N%2==1"本用于计算数字 N 是否为奇数，但是开发者没有考虑到负数的情况，从而导致该算法失败，因为任何负数应用这个算法都会等于-1。

实例 080　生成没有规律的编号

光盘位置：光盘\MR\03\080

中级

实用指数：★★☆

实例说明

本实例对实例 079 作了修改，生成的是没有规律的编号。该编号是由 3 位的随机数加上 10 位的整型随机数和一个 19 位的长整型随机数的字符串连接而成，总长度为 32 位。采用这样的方法可避免生成的编号出现重复的情况。本实例运行结果如图 3.23 所示。

图 3.23　无规律编号

关键技术

本实例通过 Random 类获取随机数，来生成没有规律的编号。该类的构造方法如下。

- Random()方法：创建一个新的随机数生成器。
- Random(long seed)方法：使用指定的参数，创建一个新的随机数生成器。该对象是随机数生成器内部状态的初始值。

设计过程

在项目中创建类 PersonelUtil，在该类中定义向数据库中插入数据的方法 insertPersonnel()。具体代码参见配书光盘中的源程序。当用户单击页面中的“提交”按钮时，会转发至 InsertServlet，在该 Servlet 中实现将用户添加的信息插入到数据库中。具体代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        Personnel personnel = new Personnel();                              //创建与数据表对应的 JavaBean 对象
        StringBuilder idb = new StringBuilder();                            //创建字符串缓冲对象
        Random ran = new Random(System.currentTimeMillis());                //根据当前时间的毫秒数创建随机数流
        idb.append(String.format("%019d", Math.abs(ran.nextLong()))
                    + String.format("%03d", (int) (Math.random() * 100 % 9) + 1));
        idb = idb.reverse();                                                //对字符串缓冲对象取反
        String id = idb + "" + String.format("%010d", Math.abs(ran.nextInt()));  //将字符串进行格式化
        personnel.setpId(id);                                               //设置对象的编号属性
        personnel.setpAge(Integer.parseInt(request.getParameter("ageTextfield")));  //设置对象的姓名属性
        personnel.setpName(request.getParameter("nameTextfield"));
        personnel.setpDept(request.getParameter("deptTextfield"));
        personnel.setpSex(request.getParameter("ageTextfield"));
        personnel.setShoolAge(request.getParameter("shoolageTextfield"));
        PersonelUtil util = new PersonelUtil();                             //创建包含有插入方法的 JavaBean 对象
        boolean bool = util.insertPersonnel(personnel);                     //调用插入员工信息方法
        String message = "数据添加失败！！";                                  //定义表示提示信息的字符串对象
        if(bool == true){
            message = "数据添加成功！！";
        }
        request.setAttribute("message", message);                           //将信息保存到 request 对象中
        request.getRequestDispatcher("index.jsp").forward(request, response);  //提供请求转发地址
}
```

心法领悟 080：无规律编号的其他用法。

本实例实现的是无规律编号，这样的编号不仅可以作为一张表的主键，还可以作为其他程序的唯一标识。

例如，在上传文件时，为了避免上传文件的名称冲突，可以使用无规律的编号形式。

实例 081 在插入数据时过滤危险字符

光盘位置：光盘\MR\03\081

中级

实用指数：★★☆

实例说明

为了保证项目的安全，通常会对用户添加的信息进行字符过滤，把一些危险的信息过滤掉。本实例实现的是将用户输入的注册信息中的危险数据过滤掉。例如，如果用户输入带有“%&”的信息（如图 3.24 所示），则在向数据库中插入数据时，会将“%&”字符过滤掉，如图 3.25 所示。

图 3.24　注册信息中包含“%&”

id	name	password	sex	email	address
1	mr	mrsoft	男	1221@sdf	吉林长春

图 3.25　数据库中数据

关键技术

本实例主要应用 String 类中的 replaceAll()方法实现，其语法如下：

```
replaceAll(String source,String replace)
```

replaceAll()方法的功能是替换字符。

参数说明

❶ source：要替换的字符串

❷ replace：用来替代的字符串。

设计过程

在项目中创建类 DoString，在该类中定义 dostring()方法。该方法包含一个 String 类型的参数，用于指定要进行过滤的字符串，并将过滤后的结果返回。具体代码如下：

```
public static String dostring(String checkstr){
        String str = "";                              //定义保存返回值的字符串对象
        str = checkstr.replaceAll("&","");            //替换字符处理
        str = str.replaceAll(";","");                 //进行字符串替换
        str = str.replaceAll("'","");
        str = str.replaceAll("<","");
        str = str.replaceAll(">","");
        str = str.replaceAll("--","");
        str = str.replaceAll("\"\"","");
        str = str.replaceAll("/","");
        str = str.replaceAll("%","");
        return str;                                   //返回过滤后的字符串
}
```

秘笈心法

心法领悟 081：过滤敏感词。

除了危险字符，有时还需要过滤敏感词。使用本实例提供的技术也可以实现敏感词的过滤。

实例 082　将用户选择的爱好信息以字符串形式保存到数据库

光盘位置：光盘\MR\03\082

中级

实用指数：★★☆

实例说明

在很多 Web 应用中，都会要求用户填写个人爱好信息。通常情况下，系统会以复选框的形式提供一些爱好信息供用户选择。通过 request 对象的 getParameterValues()方法，可得到字符串数组形式的爱好信息，但是在保存到数据库中时，需要将字符串数组转换成字符串的形式。本实例实现的是将用户选择的喜欢阅读的图书种类以字符串形式保存到数据库中，运行结果如图 3.26 所示。

图 3.26　将用户选择的爱好信息以字符串形式保存到数据库

关键技术

本实例中实现将字符串数组转换成字符串使用了 StringBuffer 对象，通过遍历字符串数据，之后通过 StringBuffer 对象的 append()方法追加内容。append()方法提供了多种重载形式，可以向 StringBuffer 对象中追加任意类型的数据。

- append(boolean bool)：将 boolean 参数的字符串表示形式追加到序列。
- append(char ch)：将 char 参数的字符串表示形式追加到此序列。
- append(string str)：将指定的字符串追加到此字符序列。

要将 StringBuffer 对象转换成 String 对象，只需调用 StringBuffer 类的 toString()方法即可。

设计过程

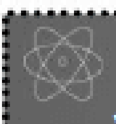
（1）在项目中创建类 LikesUtil，在该类中定义连接数据库的方法、向数据表中插入内容的方法。具体代码参见配书光盘中的源程序。

（2）当用户完成信息添加后，系统将提交至 LikesServlet。在该 Servlet 中，将处理用户提交的数据并保存到数据库中。具体代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        StringBuffer buffer = new StringBuffer();                        //定义 StringBuffer 对象
        LikesUtil util = new LikesUtil();                                //定义 LikesUtil 对象
        String[] likes = request.getParameterValues("checkbox");         //获取用户提交的爱好信息
        for(int i = 0;i<likes.length;i++){                               //循环遍历爱好数组
            buffer.append(likes[i]+"、");                                //向 StringBuffer 对象中追加内容
        }
        Likes like = new Likes();                                        //创建与爱好表对应的 JavaBean 对象
        String name = request.getParameter("nameTextfield");             //获取用户添加的用户名信息
        like.setName(name);
        like.setLikes(buffer.toString());                                //将 StringBuffer 对象转换成 String 对象
        util.insertUtil(like);                                           //调用数据添加方法
        request.setAttribute("message",name+"选择了"+":"+buffer.toString()); //将信息保存到 request 对象中
        request.getRequestDispatcher("index.jsp").forward(request, response); //设置请求转发地址
}
```

心法领悟 082：使用分隔符。

本实例中使用了“、”分隔符对爱好信息进行分隔，这样用户在从数据库中读取内容时，就会方便一些。当然，程序设计人员也可以根据自己的喜好选择其他的分隔符。



实例 083 实现跨数据库的表内容复制

光盘位置：光盘\MR\03\083

中级

实用指数：★★★

实例说明

将同一个数据库中的表进行数据传递很容易，但如果两张表在不同的数据库中，就需要进行一定的处理。本实例实现的是将 db_database03 数据库中的 tb_user 表复制到 db_database04 数据库中的 tb_user 表，运行结果如图 3.27 所示。

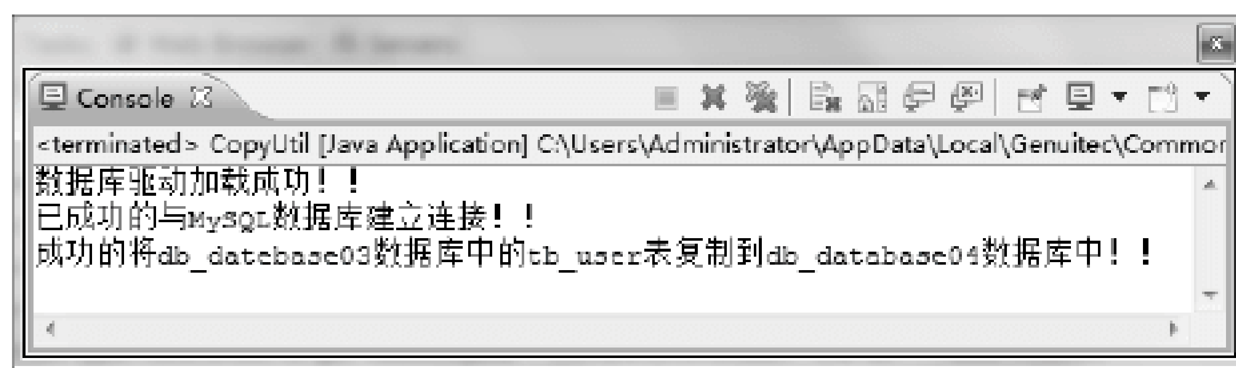


图 3.27　跨数据库的表内容复制

注意：在实现跨表的数据复制时，要保证表中字段的数据类型的一致性，否则会导致插入数据的错误。

关键技术

在实现跨数据库的表内容复制时，需要注意的是在每个数据表前要加上数据库的名称。具体语法如下：

```
数据库名.数据表
```

例如：

```
select userName,passWord from db_database03.tb_user
```

设计过程

新建项目，在该项目中新建 CopyUtil 类，在该类中定义连接数据库方法。具体代码参见配书光盘中的源程序，接下来，在该类中定义跨数据库的表内容复制方法 insertDate()。具体代码如下：

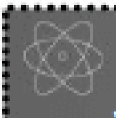
```
public void insertDate(){
        conn = getConnection();                              //获取数据库连接
        try {
            Statement statement = conn.createStatement();    //获取 Statement 对象
            //定义跨数据库的表内容复制 SQL 语句
            String sql = "insert into db_database04.tb_user(userName,passWord) select userName,passWord from db_database03.tb_user";
            statement.executeUpdate(sql);                    //执行 SQL 语句
            System.out.println("成功的将 db_datebase03 数据库中的 tb_user 表复制到 db_database04 数据库中！！");        //输出提示信息
        } catch (Exception e) {
            e.printStackTrace();
        }
}
```

秘笈心法

心法领悟 083：使用 INSERT SELECT 语句应遵循的原则。

本实例使用了 INSERT SELECT 语句来实现跨数据库的表内容复制，在使用该语句时，需要遵循以下原则。

- 用 SELECT 语句选择数据时，不能从被插入数据的表中选择行。
- 在指定插入的表后所包含的字段数目必须与 SELECT 语句中返回的字段数目相同。
- 在指定插入的表后所包含的字段数据类型必须与 SELECT 语句中返回的字段数据类型对应或系统可以自动转换。



实例 084 使用 UNION ALL 语句批量插入数据

光盘位置：光盘\MR\03\084

高级

实用指数：★★☆

实例说明

实现批量向数据表中插入数据是很常见的一种功能，实现这一功能可以使用多种方法，如 JDBC 中就有实

现批处理的类（后面章节将详细介绍关于批处理的实例）。

本实例将使用 UNION ALL 语句实现批量插入数据，插入前与插入后 tb_stu 表中数据的变化如图 3.28 所示。

id	name	shoolAge
1	双双	本科
2	李丽	硕士研究生
3	小雨	本科
4	小兰	大专

id	name	shoolAge
1	双双	本科
2	李丽	硕士研究生
3	小雨	本科
4	小兰	大专
5	王爽	硕士研究生
6	朱莉	本科
7	小宁	本科

图 3.28　本实例运行前后数据库中数据的变化

关键技术

本实例使用 UNION ALL 语句实现批处理，其语法如下：

```
INSERT tableName
SELECT columnValue,…
UNION ALL
SELECT columnValue,…
```

参数说明

❶ tableName：要添加数据的数据表。

❷ columnValue：要添加到数据表中的数据。

设计过程

在项目中创建类 StruUtil，在该类中定义连接数据库的方法 getConnection()。具体代码参见配书光盘中的源程序，在该类的主方法中编写代码，实现向 tb_stu 表中批量添加数据。具体代码如下：

```
public static void main(String[] args) {
        conn = getConnection();                                      //获取数据库连接
        try {
                Statement statement = conn.createStatement();        //获取 Statement 对象
                String sql = "insert tb_stu(name,shoolAge) select  '王爽','硕士研究生' "
                                + "union all select '朱莉','本科' "
                                + "union all select '小宁','本科'";  //定义跨数据库的表内容复制 SQL 语句
                statement.executeUpdate(sql);                        //执行 SQL 语句
                System.out
                        .println("数据批量插入成功！！");            //输出提示信息
        } catch (Exception e) {
                e.printStackTrace();
        }
}
```

秘笈心法

心法领悟 084：UNION 运算符的使用准则。

本实例中用到了 UNION 运算符，其使用准则如下：

- ❑ 在使用 UNION 运算符组合的语句中，所有选择列表的表达式数目必须相同（列名、算术表达式、聚合函数等）。
- ❑ 使用 UNION 运算符组合的结果集中的相应列或个别查询中使用的任意列的子集必须具有相同的数据类型，并且两种数据类型之间必须存在可能的隐性数据转换，或提供了显式转换。例如，在 datetime 数据类型的列和 binary 数据类型的列之间不可能存在 UNION 运算符，除非提供了显式转换；而在 money 数据类型的列和 int 数据类型的列之间可以存在 UNION 运算符，因为它们可以进行隐性转换。
- ❑ 使用 UNION 运算符组合的各语句中，对应的结果集列出现的顺序必须相同，因为 UNION 运算符是按照各个查询给定的顺序逐个比较各列。

实例 085	更新指定记录 光盘位置：光盘\MR\03\085	中级 实用指数：★★★

在程序开发中，实现对数据的更新是一项很常见的操作。实现数据的更新时需要注意的是，首先要将更新的数据显示出来，再进行更新。本实例实现的是将员工表中的数据进行更新操作。首先将员工表中的所有数据

查询出来，如图 3.29 所示；用户可选择需要更改的内容，如图 3.30 所示。

修改员工信息				
姓名	年龄	性别	部门	修改
mr	32	男	Java部门	
小双	25	女	质量部	
小勺	23	女	基础部	
小葛	31	男	策划部	
小伟	28	男	基础部	

图 3.29　所有员工信息

修改制定员工信息	
姓名：	小双
年龄：	25
性别：	女
部门：	质量部
电话：	111
备注：	质量测试
	确定修改

图 3.30　修改指定员工信息

关键技术

本实例在实现数据更新时，使用了 UPDATE 语句。该语句的具体语法如下：

```
UPDATE  数据表名  SET  字段名  =  新的字段值  WHERE  条件表达式
```

注意： 在使用插入语句和更新语句时，如果插入或者更新的值是字符串，注意要使用单引号将值括起来。

例如，将员工 tb_emp 表中编号为 2 的员工姓名修改为“葛雷”，代码如下：

```
update tb_emp set name = '葛雷' where id = 2;
```

使用 Statement 实例的 executeUpdate()方法可实现通过 Java 程序向数据库发送修改 SQL 语句。

设计过程

（1）在项目中创建类 UpdateUtil，在该类中定义操作数据库的各种方法，然后定义按照指定编号查询用户的方法 getEmp()，该方法返回值为查询出来的与用户表对应的 JavaBean 对象。具体代码如下：

```
public Emp getEmp(int id){
    Emp emp = new Emp();                                              //定义与数据表对应的 JavaBean 对象
    conn = getConnection();                                           //获取数据库连接
    try {
        Statement    statement = conn.createStatement();
        ResultSet rest = statement.executeQuery("select * from tb_emp where id ="+id);   //定义按指定编号查询数据方法
        while(rest.next()){                                           //循环遍历查询结果集
            emp.setName(rest.getString(2));                           //根据查询结果设置 JavaBean 属性
            emp.setAge(rest.getInt(3));
            emp.setSex(rest.getString(4));
            emp.setDept(rest.getString(5));
            emp.setPhone(rest.getString(6));
            emp.setRemark(rest.getString(7));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return emp;                                                       //返回查询结果
}
```

（2）在 UpdateUtil 类中创建修改员工信息的方法 updateEmp()，该方法以 Emp 对象为参数。关键代码如下：

```
public void updateEmp(Emp emp){
    conn = getConnection();                                           //获取数据库连接
    try {
        PreparedStatement statement = conn.prepareStatement("update tb_emp set name = ? and age =? and sex = ? dept = ? and phone = ? and
                remark = ? where id = ?");                            //定义更新 SQL 语句
        statement.setString(1, emp.getName());                        //设置预处理语句参数
        statement.setInt(2, emp.getAge());
        statement.setString(3, emp.getSex());
        statement.setString(4, emp.getDept());
        statement.setString(5, emp.getPhone());
        statement.setString(5, emp.getRemark());
        statement.setInt(6, emp.getId());
        statement.executeUpdate();                                    //执行预处理语句实现更新操作
```

```
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 085：在程序开发中注意加法运算符与字符串的连接。

在输出语句中，经常为输出的数字添加一个描述前缀，例如“他的年龄是：45”。但是如果 45 是一个数学加法的公式，那么很容易出错。首先第一个数字与字符串会通过“+”符号实现字符串连接，而其后的所有数字加法运算都会被看作字符串的连接操作。解决方法是把所有数字加法用括号括起来。

实例 086　将数据表清空

光盘位置：光盘\MR\03\086

中级

实用指数：★★☆

实例说明

要实现对数据表中的数据进行删除操作，可以使用 DELETE 语句，也可以使用 TRUNCATE TABLE 语句。本实例应用 TRUNCATE TABLE 语句将用户所选数据表中的内容清空，运行结果如图 3.31 所示。

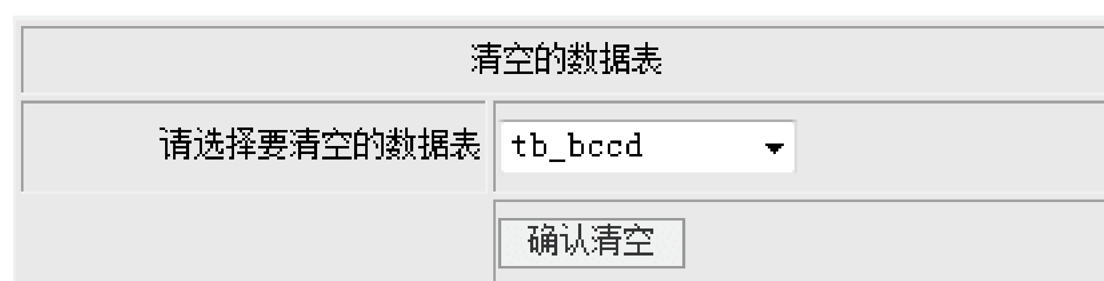


图 3.31　将数据表清空

关键技术

TRUNCATE TABLE 在功能上与不带 WHERE 子句的 DELETE 语句相同，二者均删除表中的全部行；但 TRUNCATE TABLE 比 DELETE 速度快，且使用的系统和事务日志资源少。DELETE 语句每次删除一行，并在事务日志中为所删除的每一行做一项记录；TRUNCATE TABLE 则通过释放存储表数据所用的数据页来删除数据，并且只在事务日志中记录页的释放，减少了日志空间的占用。

TRUNCATE TABLE 语句的语法如下：

```
TRUNCATE TABLE 数据表名
```

设计过程

（1）在项目中创建类 ClearUtil，在该类中定义连接数据库的方法 getConnection()、获取数据库中所有数据表的方法 listDB()。具体代码参见配书光盘中的源程序。

（2）在 ClearUtil 类中定义 deleteDate()方法，用于实现清空数据库中的指定数据表。该方法包含一个 String 类型的参数，用于指定要删除的数据表。具体代码如下：

```
public void deleteDate(String dataName){
    conn = getConnection();                                          //获取数据库连接
    try {
        Statement statement = conn.createStatement();                //获取 Statement 对象
        statement.executeUpdate("TRUNCATE TABLE  "+dataName);        //指定删除语句
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```


心法领悟 086：Java 中异常的继承层次。

Java 中所有异常对象都是 Throwable 类的子类。该类的直接子类是 Error 和 Exception。Error 代表 Java 虚拟机错误等严重的问题，通常不该出现在程序中；Exception 是需要更加关心的异常，其直接子类包括 RuntimeException 等。

第 4 章

SQL 语句应用技术

- 聚集函数与日期查询
- 排序与分组函数的应用
- 比较大小与逻辑应用

4.1　聚集函数与日期查询

实例 087　利用 SUM 函数实现数据汇总

光盘位置：光盘\MR\04\087

中级

实用指数：★★☆

实例说明

计算诸如求平均值和总和的函数称为聚集函数，在程序开发中得到了广泛的应用。本实例应用聚集函数中的求和函数 SUM，求学生成绩表中各科成绩的总和，运行结果如图 4.1 所示。

利用聚合函数SUM对学生成绩进行汇总>>>

学生成绩汇总

编号	姓名	语文	数学	英语	政治
1	王丽	92.0	91.0	85.0	91.0
2	小葛	96.0	91.0	84.0	80.0
3	小爽	91.0	86.0	94.0	84.0
4	小明	89.0	86.0	91.0	92.0
5	小丽	89.0	78.0	92.0	98.0
汇总		457.0	432.0	446.0	445.0

图 4.1　利用 SUM 函数对学生成绩进行汇总

关键技术

SUM 聚集函数用于返回表达式中所有值的和，但需要注意的是该函数只能用于数据类型的数字列，在统计的过程中 null 值将被忽略。

SUM 函数的具体语法如下：

```
SUM([ALL | DISTINCT] expression)
```

参数说明

❶ ALL：对所有的值进行聚集函数运算。ALL 是默认设置。

❷ DISTINCT：指定 SUM 返回唯一值的和。

❸ expression：是常量、列或函数，或者是算术、按位与字符串等运算符的任意组合。如果 expression 是精确数字或近似数字数据类型分类（bit 数据类型除外）的表达式，则不允许使用聚集函数和子查询。

在项目中创建类 JDBCUtil，在该类中定义进行汇总查询的方法 getSum()，该方法以 List 集合形式返回查询结果。具体代码如下：

```
public List getSum(){
    List list = new ArrayList();                        //定义保存查询结果的 List 对象
    //定义查询 SQL 语句
    String sql = "select sum(math) as smath,sum(chinese) as schinese,sum(english) as senglish ,sum(politics) as spolitics from tb_grade";
    conn = getConnection();                             //调用创建数据库连接方法
    try {
        Statement statement = conn.createStatement();   //获取 Statement 对象
        ResultSet rest = statement.executeQuery(sql);   //执行查询语句，获取查询结果集
        while(rest.next()){                             //循环遍历查询结果集
            Grade grade = new Grade();                  //创建与数据表对应的 JavaBean 对象
            grade.setChinese(rest.getFloat("schinese")); //设置对象属性
            grade.setEnglish(rest.getFloat("senglish"));
            grade.setMath(rest.getFloat("smath"));
            grade.setPolitics(rest.getFloat("spolitics"));
            list.add(grade);                            //将对象保存到 List 集合中
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return list;                                        //返回保存查询结果的 List 集合对象
}
```

秘笈心法

心法领悟 087：SUM 函数应用的数据类型。

SUM 函数只能用于 int、smallint、tinyint、decimal、numeric、float、real、money 和 smallmoney 等数据类型。

在使用 SUM 函数时，SQL Server 把数据类型 smallint 或 tinyint 当作 int 数据类型处理。



实例 088 利用 AVG 函数实现计算平均值

光盘位置：光盘\MR\04\088

中级

实用指数：★★

实例说明

如果要查询数据表中某列数据的平均值，可以使用 AVG 函数。本实例将应用 AVG 函数，计算出学生成绩表中各科学生成绩的平均值，运行结果如图 4.2 所示。

利用AVG函数计算平均值>>>

求学生成绩平均值

编号	姓名	语文	数学	英语	政治
1	王丽	92.0	91.0	85.0	91.0
2	小葛	96.0	91.0	84.0	80.0
3	小爽	91.0	86.0	94.0	84.0
4	小明	89.0	86.0	91.0	92.0
5	小丽	89.0	78.0	92.0	98.0
平均值		91.4	86.4	89.2	89.0

图 4.2　利用 AVG 函数计算平均值

关键技术

AVG 函数用于计算数据表中指定列的平均值，空值将被忽略。

语法：

```
AVG ([ALL|DISTINCT] expression )
```

参数说明

❶ ALL：对所有的值进行聚集函数运算。ALL 是默认设置。

❷ DISTINCT：指定 AVG 操作只使用每个值的唯一实例，而不管该值出现了多少次。

❸ expression：是常量、列或函数，或者是算数、按位与字符串等运算符的任意组合。例如，对学生成绩表中语文列求平均值，表达式可写成 avg(chinese)。如果 expression 是精确数字或近似数字数据类型分类的表达式（bit 数据类型除外），则不允许使用聚集函数和子查询。

设计过程

在项目中创建类 JDBCUtil，在该类中定义用于查询平均值的 getAvg()方法，该方法以 List 集合作为返回值。具体代码如下：

```
public List getAvg(){
	List list = new ArrayList();                          //定义保存查询结果的 List 对象
	//定义查询 SQL 语句
	String sql = "select avg(math) as smath,avg(chinese) as schinese,avg(english) as senglish ,avg(politics) as spolitics from tb_grade";
	conn = getConnection();                               //调用创建数据库连接方法
	try {
		Statement statement = conn.createStatement();     //获取 Statement 对象
		ResultSet rest = statement.executeQuery(sql);     //执行查询语句，获取查询结果集
		while(rest.next()){                               //循环遍历查询结果集
			Grade grade = new Grade();                    //创建与数据表对应的 JavaBean 对象
			grade.setChinese(rest.getFloat("schinese"));  //设置对象属性
			grade.setEnglish(rest.getFloat("senglish"));
			grade.setMath(rest.getFloat("smath"));
			grade.setPolitics(rest.getFloat("spolitics"));
			list.add(grade);                              //将对象保存到 List 集合中
		}
	} catch (SQLException e) {
		e.printStackTrace();
	}
	return list;                                          //返回保存查询结果的 List 集合对象
}
```

秘笈心法

心法领悟 088：实现对 List 集合按指定顺序排序。

本章中的查询结果集大多数都是以 List 集合对象的形式进行存储。List 集合中的对象是有一定的顺序的，其顺序就是对象插入集合时的顺序。那么如何实现对集合中的对象按照特定的顺序进行排序呢？

实现对 List 集合中的对象进行排序，可以使用 Collections 类的 sort()方法。例如：

```
ArrayList<Integer> alist = new ArrayList<Integer>();
alist.add(new Integer(1));
alist.add(new Integer(10));
alist.add(new Integer(5));
System.out.println(list.toString());
Collections.sort(alist);
System.out.println(list.toString());
```

实例 089　利用 MIN 函数求数据表中的最小数据

光盘位置：光盘\MR\04\089

中级

实用指数：★★☆

实例说明

聚集函数中的 MIN 函数用于查询数据表中的最小值。本实例实现的是查询订单金额最小的销售代表信息，运行结果如图 4.3 所示。

利用MIN函数求最小值>>>

查询订单金额最小的销售代表信息

编号	姓名	地址	金额	签订日期
2	王五	沈阳大连	30000.0	2010-11-20

图 4.3　利用 MIN 函数求最小值

关键技术

MIN 聚集函数主要用于返回在某一集合中对数值表达式求得的最小值。语法如下：

```
MIN ([ALL|DISTINCT] expression)
```

参数说明

❶ ALL：该参数是默认设置，如果没有该参数，将对所有的值进行聚集函数运算。

❷ DISTINCT：指定每个唯一值都被考虑。DISTINCT 对 MIN 无意义。

❸ expression：该参数为表达式，由常量、列名、函数以及算术运算符、按位运算符和字符串运算符任意组合而成。

在项目中创建类 OrderUtil，在该类中定义获取订单金额最小的销售代表信息的 getMin()方法，该方法以 List 集合作为返回值。具体代码如下：

```
public List getMin(){
	List list = new ArrayList();								//定义保存查询结果的 List 对象
	String sql = "select * from tb_order where money in(select min(money) from tb_order)";	//定义查询 SQL 语句
	conn = getConnection();								//调用创建数据库连接方法
	try {
		Statement statement = conn.createStatement();				//获取 Statement 对象
		ResultSet rest = statement.executeQuery(sql);				//执行查询语句，获取查询结果集
		while(rest.next()){								//循环遍历查询结果集
			Order order = new Order();						//定义与数据表对应的 JavaBean 对象
			order.setId(rest.getInt(1));
			order.setName(rest.getString(2));
			order.setAddress(rest.getString(3));
			order.setMoney(rest.getFloat(4));
			order.setoDate(rest.getString(5));
			list.add(order);							//向集合中添加对象
		}
	} catch (SQLException e) {
```

```
        e.printStackTrace();
        }
        return list;                                                         //返回保存查询结果的 List 集合对象
}
```

秘笈心法

心法领悟 089：MIN 函数应用的数据类型。

MIN 函数可以用于数据类型为数字、字符、datatime 的列，但是不能用于数据类型为 bit 的列，不能使用聚集函数和子函数。当使用 CUBE 或 ROLLUP 时，不支持区分聚集。例如，如果使用 MIN(DISTINCT column_name)，则 SQL Server 将返回错误信息并取消查询。

实例 090 利用 MAX 函数求数据表中的最大值

中级

光盘位置：光盘\MR\04\090

实用指数：★★★★

实例说明

利用 MAX 函数可求出数据表中指定列中的最大值。本实例将应用 MAX 函数查询订单表中签订最大金额订单的销售代表信息，运行结果如图 4.4 所示。

利用MAX函数求最大值>>>

查询订单金额最大的销售代表信息

编号	姓名	地址	金额	签订日期
5	小碗	北京	540000.0	2010-10-12

图 4.4 查询订单金额最大的销售代表信息

关键技术

MAX 聚集函数主要用于返回在某一集合中对数值表达式求得的最大值。

语法：

```
MAX([ALL|DISTINCT] expression)
```

参数说明

❶ ALL：该参数是默认设置；如果没有该参数，将对所有的值进行聚集函数运算。

❷ DISTINCT：指定每个唯一值都被考虑。DISTINCT 对 MAX 无意义。

❸ expression：该参数为表达式，由常量、列名、函数以及算术运算符、按位运算符和字符串运算符任意组合而成。

在项目中创建类 OrderUtil，在该类中定义查询最大值方法 getMAX()（该方法包含一个 List 类型的返回值），并保存查询结果。具体代码如下：

```
public List getMAX(){
        List list = new ArrayList();                                         //定义保存查询结果的 List 对象
        String sql = "select * from tb_order where money in(select max(money) from tb_order)";  //定义查询 SQL 语句
        conn = getConnection();                                              //调用创建数据库连接方法
        try {
                Statement statement = conn.createStatement();                //获取 Statement 对象
                ResultSet rest = statement.executeQuery(sql);                //执行查询语句获取查询结果集
                while(rest.next()){                                          //循环遍历查询结果集
                        Order order = new Order();                           //定义数据表对应的 JavaBean 对象
                        order.setId(rest.getInt(1));
                        order.setName(rest.getString(2));
                        order.setAddress(rest.getString(3));
                        order.setMoney(rest.getFloat(4));
                        order.setoDate(rest.getString(5));
                        list.add(order);                                     //向集合中添加对象
                }
        } catch (SQLException e) {
                e.printStackTrace();
```

```
        }
        return list;                                                    //返回保存查询结果的 List 集合对象
}
```

秘笈心法

心法领悟 090：利用 break 避免死循环。

充分利用循环可以提高程序的开发与执行效率，但是如果不注重循环中的算法，则很容易导致程序的死循环。因此，在循环体中要对可能出现的特殊情况使用 break 语句中断循环。

实例 091　利用 COUNT 函数求销售额大于某值的图书种类　中级

光盘位置：光盘\MR\04\091　实用指数：★★☆

实例说明

COUNTN 函数用于返回表达式中值的个数，在统计计算中发挥着重要的作用。本实例实现的是应用 COUNT 函数查询销售额大于某值的图书种类信息，运行结果如图 4.5 所示。

count函数查询图书种类

查询销售额超过：300 的图书种类　查询　有2种

编号	图书名称	销量	销售日期
S001	Java程序开发完成自学手册	450	2010/7/8
S002	Java从入门到精通	360	2010/12/4
S003	JavaWeb开发实战宝典	490	2010/11/26
S004	JavaWeb典型模块大全	580	2010/10/12
S005	Jsp程序开发案例全程实录	610	2010/9/6

图 4.5　本实例的运行结果

关键技术

COUNT 函数能够计算在一条记录中表达式的数目，即数据条数。一般地，COUNT 函数主要用于查询结果中的数据条数，因此通常以通配符“*”作为该函数的参数。事实上，COUNT 函数也是唯一允许使用通配符作为参数的聚集函数。

语法：

```
COUNT({[ALL | DISTINCT]expression} | *)
```

参数说明

❶ ALL：该参数是默认设置；如果没有该参数，SQL Server 将对所有的值进行聚集函数运算。

❷ DISTINCT：该参数指定 COUNT 函数返回唯一非空值的数量。

❸ expression：该参数为表达式，其类型可以是除 uniqueidentifier、text、image 或 ntext 之外的任何类型。

❹ *：指定应该计算所有行以返回表中行的总数。COUNT(*)不需要任何参数，而且不能与 DISTINCT 一起使用。COUNT(*)不需要 expression 参数，因为根据定义，该函数不使用有关任何特定列的信息。COUNT(*)返回指定表中行的数量而不消除副本，对每行分别进行计数，包括含有空值的行。

在项目中创建类 BookUtil，在该类中定义操作数据库的相关方法，然后定义获取销售额大于某值的图书种类方法 getCount()（该方法包含一个 int 类型的参数，用于指定要查询的条件），并以 int 类型返回查询的结果。具体代码如下：

```
public int getCount(int total){
        int count = 0;
        conn = getConnection();                                         //调用获取数据库连接方法
        try {
                Statement statement = conn.createStatement();           //获取 Statement 对象
                String sql = "select count(bookName)as count from tb_bookSell where total >"+total;   //定义查询 SQL 语句
                ResultSet rest = statement.executeQuery(sql);           //执行查询语句获取查询结果集
                while(rest.next()){                                     //循环遍历查询结果集
                count = rest.getInt("count");                           //获取查询结果
                }
        } catch (Exception e) {
                e.printStackTrace();
        }
```

```
        return count;
}
```

秘笈心法

心法领悟 091：使用日期格式器。

日期字符串的格式因语言环境而不同，虽然可以通过灵活的字符串操作手动实现指定环境的日期格式，但还是建议采用 Java 提供的日期格式器来完成，这样可以提高程序开发速度，而且采用已有组件可以避免调试与错误的发生。

实例 092　查询与张静同一天入司的员工信息

光盘位置：光盘\MR\04\092

中级

实用指数：★★☆

实例说明

本实例实现的是查询与张静同一天入司的员工信息并显示在页面中。可以使用多种方法来实现，本实例使用的是 CONVERT 函数与 LIKE 关键字，CONVERT 函数可以对指定的日期进行格式化，LIKE 关键字可以进行模糊查询。本实例运行结果如图 4.6 所示。

显示统计结果>>>

与张静同一天入司的员工信息

姓名	性别	部门	入司时间
小许	男	Java开发部	2007/5/15
张静	女	质量部	2007/5/15
小葛	男	系统分析	2007/5/15

图 4.6　查询与张静同一天入司的员工信息

本实例在对日期型数值进行模糊查询时，应用了 CONVERT 函数。该函数为日期格式化函数，可将日期转换成 yyyy-mm-dd 等格式。该函数语法如下：

```
CONVERT(date_type[(length)] , expression , style)
```

参数说明

❶ date_type：要转换的数据类型。

❷ expression：DATETIME 类型的数据。

❸ style：指定转换形式。其取值不同，对应的日期、时间格式也不同。如表 4.1 所示为 style 取不同值时所对应的日期、时间格式。

表 4.1　style 不同取值对应的日期、时间格式

style	格　　式	示　　例
0	mon dd yyyy hh:miAM/PM	Jun 22 2009 08:30 AM
1	mm/dd/yy	08/28/09
2	yy.mm.dd	09.04.24
3	dd/mm/yy	02/12/09
4	dd.mm.yy	22.06.09
5	dd-mm-yy	22-07-09
6	dd mon yy	21 Jun 09
7	Mon dd yy	Jun 21 09
8	hh:mm:ss	11:30:21
9	mon dd yyyy hh:mi:ss:mmmmAM/PM	Jun 10 2009 08:25:25:048 AM
10	mm-dd-yy	07-25-09
11	yy/mm/dd	09/07/24
12	yymmdd	090712
13	dd mon yyyy hh:mm:ss:mmm	21 Jun 2009 09:20:22:045

续表

style	格　　式	示　　例
14	hh:mi:ss:mmm	09:31:22:048
20	yyyy-mm-dd hh:mi:ss	2009-07-25 08:25:20

本实例在 SELECT 查询语句中使用了 LIKE 关键字。在查询语句中，LIKE 关键字位于 WHERE 子句中。该关键字可以与 NOT 运算符组合使用。在查询语句中使用 LIKE 关键字的语法如下：

```
SELECT [DISTIN|UNIQUE](*,columname[AS alias],…)
FROM table
WHERE columname LIKE '[ _ | % ]value'
```

参数说明

❶ columname：要进行查询匹配的字段。

❷ value：要进行匹配的字符串。

❸ 匹配符“_”表示任意单个字符，只能匹配一个字符；匹配符“%”可以匹配 0 个或更多字符的任意长度的字符串，且不考虑字符的多少。

设计过程

在项目中定义 getList()方法，在该类中定义查询与张静同一天入司的员工信息方法 getList()，该方法以 List 集合返回查询结果。具体代码如下：

```
public List getList(){
    List list = new ArrayList();                                    //定义保存查询结果的 List 对象
    conn = getConn();                                               //获取数据库连接
    try {
        Statement statement = conn.createStatement();               //获取 Statement 对象
        String sql = "select * from tb_emp where convert(varchar(10),ddate,21)" +
                "like (select convert(varchar(10),ddate,21) from tb_emp where name = '张静')";   //定义查询与张静同一天入司的员工信息
        ResultSet rest = statement.executeQuery(sql);               //执行查询语句返回查询结果集
        while(rest.next()){                                         //循环遍历查询结果集
            Emp emp = new Emp();                                    //定义与数据库表对应的 JavaBean 对象
            emp.setId(rest.getInt(1));                              //设置对象属性
            emp.setName(rest.getString(2));
            emp.setSex(rest.getString(3));
            emp.setDept(rest.getString(4));
            emp.setDdate(rest.getString(5));
            list.add(emp);                                          //向集合中添加对象
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return list;
}
```

秘笈心法

心法领悟 092：注意模糊查询中的通配符。

在使用 LIKE 关键字进行模糊查询时，需要注意 LIKE 关键字给定的表达式中的所有字符都是有效的，包含开头和结尾中的空格。

实例 093　使用 IN 谓词查询某几个时间的数据

中级

光盘位置：光盘\MR\04\093

实用指数：★★★★☆

实例说明

应用 IN 谓词可以在查询中指定多个条件，如本实例实现的是使用 IN 谓词查询在 2008/10/5、2010/4/5、2007/

5/15 入司的员工信息，运行结果如图 4.7 所示。

显示查询结果>>>

查询在'2008/10/5','2010/4/5','2007/5/15'入司的员工信息

姓名	性别	部门	入司时间
小双	女	Java开发部	2008/10/5
小许	男	Java开发部	2007/5/15
小兰	女	VB开发部	2010/4/5
张静	女	质量部	2007/5/15
小葛	男	系统分析	2007/5/15

图 4.7　查询在 2008/10/5、2010/4/5、2007/5/15 入司的员工信息

■ 关键技术

应用 IN 谓词可确定给定的值是否与子查询或列表中的值相匹配，常用于引入子查询。

语法：

```
test_expression [ NOT ] IN
    (
        subquery
        | expression [ ,...n ]
    )
```

参数说明

❶ test_expression：任何有效的 SQL 表达式。

❷ subquery：包含某列结果集的子查询，该列必须与 test_expression 具有相同的数据类型。

❸ expression [,...n]：一个表达式列表，用来测试是否匹配。所有的表达式必须和 test_expression 具有相同的数据类型。

如果 test_expression 与 subquery 返回的任何值相等，或与逗号分隔的列表中的任何 expression 相等，那么结果值为 true，否则结果值为 false。

NOT 关键字可反转 IN 测试的结果。如果用 NOT IN 引入子查询，并在子查询的结果表中没有匹配的项，则执行 SQL 语句指定的动作。例如，下面的 SQL 语句用于查询 10 月份工资不在 1500～2000 元范围内的员工信息。

```
select * from tb_mrgzslb where salary not in (select salary from tb_mrgzslb where salary between 1500 and  2000) and salarymonth='10''
```

■

（1）在项目中创建类 EmpUtil，在该类中定义查询在 2008/10/5、2010/4/5、2007/5/15 入司的员工信息的方法，并以 List 集合返回查询结果。具体代码如下：

```
public List getList(){
        List list = new ArrayList();                                          //定义保存查询结果的 List 对象
        conn = getConn();                                                     //获取数据库连接
        try {
            Statement statement = conn.createStatement();                     //获取 Statement 对象
            String sql = "select * from tb_emp where ddate in ('2008/10/5','2010/4/5','2007/5/15')";   //定义查询 SQL 语句
            ResultSet rest = statement.executeQuery(sql);                     //执行查询语句返回查询结果集
            while(rest.next()){                                               //循环遍历查询结果集
                Emp emp = new Emp();                                          //定义与数据库表对应的 JavaBean 对象
                emp.setId(rest.getInt(1));                                    //设置对象属性
                emp.setName(rest.getString(2));
                emp.setSex(rest.getString(3));
                emp.setDept(rest.getString(4));
                emp.setDdate(rest.getString(5));
                list.add(emp);                                                //向集合中添加对象
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return list;
}
```

（2）在 list.jsp 页面中定义表格，显示查询结果。具体代码如下：

```
<table width="353" height="65" border="1" align="center">
    <tr>
```

```
        <td><div align="center">姓名</div></td>
        <td><div align="center">性别</div></td>
        <td><div align="center">部门</div></td>
        <td><div align="center">入司时间</div></td>
    </tr>
    <%
        List list = (List)request.getAttribute("list");                    //获取保存在 request 对象中的对象
        for(int i = 0;i<list.size();i++){                                   //循环遍历查询结果
                Emp emp = (Emp)list.get(i);                                 //获取结果集中对象
    %>
    <tr>
        <td><div align="center"><%=emp.getName()%></div></td>               //将对象相关属性显示在页面中
        <td><div align="center"><%=emp.getSex() %></div></td>
        <td><div align="center"><%=emp.getDept()%></div></td>
        <td><div align="center"><%=emp.getDdate() %></div></td>
    </tr>
    <%} %>
</table>
```

秘笈心法

心法领悟 093：使用 get()方法与 post()方法传递参数。

在 URL 地址中可以附加一些参数，每个参数由参数名和参数值组成。在参数传递中可以使用 get()方法，也可以使用 post()方法。使用 get()方法传递参数时，参数部分将附加在请求行中的资源路径后面；使用 post()方法请求参数时，是将数据作为 HTTP 消息的实体内容发送给 Web 服务器，而不是作为 URL 地址的参数传递。

实例 094　对数据进行降序排序查询

光盘位置：光盘\MR\04\094

中级

实用指数：★★☆

实例说明

本实例实现将图书库存表中的图书信息按图书价格或者图书数量进行降序排列。在“请选择排序字段：”右侧的下拉列表框中选择“按图书价格”或“按图书数量”选项，单击“降序排序”按钮，即可将图书价格或者图书数量按照降序排列并显示在下面的表格中，如图 4.8 所示。

图书数量统计

请选择排序字段：　按图书价格　降序排序

图书名称	图书价格	图书作者	图书数量	出版社	存放位置
Java从入门到精通（第二版）	69元	明日科技	10册	机械出版社	1号仓库
Asp.net从入门到精通（第二版）	68元	明日科技	15册	机械出版社	2号仓库
Jsp项目开发案例全程实录	67元	明日科技	23册	电子出版社	3号仓库
C语言从入门到精通	66元	明日科技	14册	机械出版社	5号仓库
C++从入门到精通	63元	明日科技	26册	电子出版社	4号仓库
C#从入门到精通（第二版）	56元	明日科技	12册	电子出版社	1号仓库

图 4.8　对数据进行降序排序查询

要实现对数据进行降序排列查询，需在 SQL 语句中使用 ORDER BY 子句和 DESC 关键字。其实现方法是在 SQL 语句的查询语句中添加 order by price desc 或者 order by store desc 子句。

ORDER BY 子句的作用是分类输出，并且根据表中包含的一列或多列表达式将输出的值按升序或降序排列。它不改变数据库中行的顺序，只是简单地改变查询输出的值的显示顺序。其语法格式如下：

```
SELECT …
WHERE …
ORDER BY expression [ASC|DESC],…
```

参数说明

❶ expression：一个表达式，通常是一个输出时用来分类的字段。

❷ [ASC|DESC]：可选项，代表升序或降序。

❸ ,…：指可以有多于一个的分类表达式，并且每一个表达式都可以为升序或降序。

排序表达式中可包括未出现在 SELECT 子句选择列表中的列名。如果在 SELECT 子句中使用了 DISTINCT 关键字，或查询语句中包含 UNION 运算符，则排序列必须包含在 SELECT 子句选择列表中。

设计过程

（1）创建数据库连接类 DBbean，主要是通过加载驱动程序、建立数据库连接的步骤得到 Connection 类的一个对象。

（2）在 JSP 页面中通过 JSP 的 useBean 标记获取 DBbean 类中的连接，利用 request 对象的 getParameter() 方法获得下拉列表框中选择的数据，把得到的数据作为 SQL 语句的一个字段，然后通过执行 SQL 语句获得查询结果，并将查询结果输出到表格中。具体代码如下：

```
<%
        String order=request.getParameter("order");
        try{
          Connection connection=con.getCon();                        //得到一个数据库连接
          Statement st=connection.createStatement();
          String sql="select * from tb_library order by "+order+" desc ";
          ResultSet rs=st.executeQuery(sql);                         //执行 SQL 语句
          if(rs.next())
            {
            do{                                                      //显示查询结果
%>
        <tr>
          <td width="183" align="left"><div align="center"><%=rs.getString("name")%></div></td>
          <td width="60" align="left"><div align="center"><%=rs.getString("price")%>元</div></td>
          <td width="84" align="left"><div align="center"><%=rs.getString("author")%></div></td>
          <td width="66" align="left"><div align="center"><%=rs.getString("store")%>册</div></td>
          <td width="83" align="left"><div align="center"><%=rs.getString("publisher")%></div></td>
          <td width="89" align="left"><div align="center"><%=rs.getString("position")%></div></td>
        </tr>
        <%
      }while(rs.next());
      }
      else
      {
          out.println("<br><strong><center>没有你要检索的数据!</center><strong>");
      }
   } catch(Exception ex) {System.out.println(ex.getMessage());}%>
```

秘笈心法

心法领悟 094：SQL 语句中变量的使用。

在 SQL 语句中出现 String 类型的变量时，需要把该变量用“+expression+”的形式（其中 expression 表示一个变量）写入到 SQL 语句中，这样才符合 SQL 语句的规定。

实例 095 数据的多条件排序查询

光盘位置：光盘\MR\04\095

中级

实用指数：★★☆

实例说明

本实例实现的是将水果信息表中的水果信息按选择的字段进行多条件升序排列。选择第一个下拉列表框中的“按进货价”（或者“按出售价”），然后选择第二个下拉列表框中的“按生产日期”（或者“按保质期”），单击右侧的“升序排序”按钮，即可将水果信息表中的数据按进货价和按生产日期升序排列，并显示在下面的表格中，如图 4.9 所示。

数据多条件排序查询>>>

按进货价 ▾　　按生产日期 ▾　　升序排序

编号	名称	进货价	出售价	生产日期	保质期	产地	质量
1	苹果	2元	3元	2010-09-08	2010-10-09	河南	优
3	菠萝	2元	4元	2010-09-08	2010-10-08	吉林	良
5	甘蔗	2元	2元	2010-09-11	2010-10-26	云南	优
6	草莓	3元	6元	2010-09-07	2010-10-10	福建	良
2	香蕉	3元	4元	2010-09-10	2010-11-01	山东	良
4	橘子	4元	5元	2010-09-11	2010-10-24	四川	优

图 4.9　数据的多条件排序查询

要实现数据的多条件排序查询，需在 SQL 语句中使用 ORDER BY 子句和 ASC 关键字。若用户选择按进货

价和按生产日期进行升序排列，其实现方法是在 SQL 语句的查询语句中添加 order by enterprice asc,date asc 子句（该子句的作用是先按照进货价进行升序排列，若进货价相同，再按照生产日期升序排列）。其中，ASC 关键字是可以省略的，默认情况下是按升序排列。

设计过程

（1）创建数据库连接类 DBbean，主要是通过加载驱动程序、建立数据库连接的步骤得到 Connection 类的一个对象。

（2）在 JSP 页面中通过 JSP 的 useBean 标记获取 DBbean 类中的连接，利用 request 对象的 getParameter() 方法获得下拉列表中选择的数据，把得到的数据作为 SQL 语句的一个字段，然后通过执行 SQL 语句获得查询结果，并将查询结果输出到表格中。具体代码如下：

```
<%
        String sql="";
        String sel=request.getParameter("select");
        String sel2=request.getParameter("select2");
        if(sel==null&&sel2==null){
        sql="select * from tb_fruit";
        }
        else
        {
         sql="select * from tb_fruit order by "+sel+" asc,"+sel2+" asc";
        }
        try{
         Connection connection=con.getCon();                          //得到一个数据库连接
         Statement st=connection.createStatement();
         ResultSet rs=st.executeQuery(sql);                           //执行 SQL 语句
         if(rs.next())
           {
           do{                                                        //显示查询结果
%>
        <tr>
          <td width="51" align="left"><div align="center"><%=rs.getString("id")%></div></td>
          <td width="60" align="left"><div align="center"><%=rs.getString("name")%></div></td>
          <td width="65" align="left"><div align="center"><%=rs.getString("enterprice")%>元</div></td>
          <td width="63" align="left"><div align="center"><%=rs.getString("outerprice")%>元</div></td>
          <td width="70" align="left"><div align="center"><%=rs.getString("date")%></div></td>
          <td width="71" align="left"><div align="center"><%=rs.getString("fresh")%></div></td>
          <td width="49" align="left"><div align="center"><%=rs.getString("address")%></div></td>
          <td width="53" align="left"><div align="center"><%=rs.getString("quality")%></div></td>
        </tr>
        <%
      }while(rs.next());
      }
      else
      {
         out.println("<br><strong><center>没有你要检索的数据!</center><strong>");
      }
   } catch(Exception ex) {System.out.println(ex.getMessage());}%>
```

秘笈心法

心法领悟 095：多条件排序时 ORDER BY 子句的使用。

在 SQL 语句中若进行多条件排序，需要把多个排序的条件用“,”隔开。

实例 096	对统计结果进行排序 光盘位置：光盘\MR\04\096	中级 实用指数：★★✩

实例说明

本实例实现的是将商品库存表中的库存数量按照商品名称或进货地点进行统计并且排序。在“请选择汇总

字段：”右侧的下拉列表框中选择 “按进货地点”或者“按商品名称”选项，单击“库存汇总”按钮，即可将商品库存表中的库存数量按进货地点或按商品名称进行统计，并将统计结果按照升序排列显示在下面的表格中，如图 4.10 所示。

商品库存统计		
请选择汇总字段：	按进货地点 ▾	库存汇总

进货地点	库存汇总
陕西	54件
山东	57件
湖北	67件
吉林	92件
河南	146件

图 4.10　对统计结果进行排序

■ 关键技术

要实现对统计结果排序，需要在 SQL 语句中使用 ORDER BY 子句、GROUP BY 子句、SUM 聚集函数和 DESC 关键字。下面对其使用的方法进行详细的讲解。

（1）GROUP BY 子句

语法如下：

```
[GROUP BY [ALL] expression [,...n]
 [WITH{CUBE|ROLLUP}]
]
```

参数说明

❶ ALL：表示选定数据列中所有数据。如果用户指定了 ALL，将对组中不满足搜索条件的汇总列返回空值。

❷ expression：对查询执行分组的表达式。expression 也称为分组列。它可以是列或列的非聚集表达式。在选择列表内定义的列的别名不能用于指定分组列。

❸ CUBE：指定在查询的结果集内不仅包含由 GROUP BY 提供的正常行，还包括汇总行。在结果集内返回每个可能的组和子组组合的 GROUP BY 汇总行，GROUP BY 汇总行在结果中显示为 null，但可用来表示所有值。

❹ ROLLUP：指定在结果集内不仅包含由 GROUP BY 子句提供的正常行，还包含汇总行。按层次结构顺序，从组内的最低级别到最高级别汇总组。组的层次结构取决于指定分组列时所使用的顺序，更改分组列的顺序会影响在结果集内生成的行数。

（2）SUM 聚集函数

语法如下：

```
SUM ( [ALL|DISTINCT] expression )
```

参数说明

❶ ALL：对所有的值进行聚集函数运算。ALL 是默认设置。

❷ DISTINCT：指定 SUM 返回唯一值的和。

❸ expression：是常量、列或函数，或者是算术、按位与字符串等运算符的任意组合。expression 是精确数字或近似数字数据类型分类（bit 数据类型除外）的表达式，不允许使用聚集函数和子查询。

■

（1）创建数据库连接类 DBbean，主要是通过加载驱动程序、建立数据库连接的步骤得到 Connection 类的一个对象。

（2）在 JSP 页面中通过 JSP 的 useBean 标记获取 DBbean 类中的连接，利用 request 对象的 getParameter() 方法获得下拉列表中选择的数据并且进行相应的判断，根据判断结果的不同书写相应的 SQL 语句，然后执行相应的 SQL 语句获取不同的查询结果，并将查询结果输出到表格中。具体代码如下：

```
<%
    String str=request.getParameter("order");
    String sql="";
    if(str==null){                    //判断得到的数据是不是为 null，若为 null，显示所有的商品库存信息
      %>
        </div></td>
    </tr>
</table>
  <table width="596" height="21" border="1">
    <tr bgcolor="#FFCCFF">
      <td width="70"><div align="center"><strong>商品编号</strong></div></td>
```

```
        <td width="89"><div align="center"><strong>商品名称</strong></div></td>
        <td width="92"><div align="center"><strong>库存数量</strong></div></td>
        <td width="102"><div align="center"><strong>进货价钱</strong></div></td>
        <td width="111"><div align="center"><strong>进货时间</strong></div></td>
        <td width="91"><div align="center"><strong>进货地点</strong></div></td>
      </tr>
    </table>
    <table width="596" height="28" border="1">
      <%
         try{
            Connection connection=con.getCon();                              //得到一个数据库连接
            Statement st=connection.createStatement();
            sql ="select * from tb_clothes";                                 //利用 SQL 语句进行查询
            ResultSet rs=st.executeQuery(sql);                               //执行 SQL 语句
            while(rs.next()){
            %>
      <tr>
        <td width="70" align="left"><div align="center"><%=rs.getString("id")%></div></td>
        <td width="89" align="left"><div align="center"><%=rs.getString("name")%></div></td>
        <td width="92" align="left"><div align="center"><%=rs.getString("total")%>件</div></td>
        <td width="102" align="left"><div align="center"><%=rs.getString("price")%>元</div></td>
        <td width="111" align="left"><div align="center"><%=rs.getString("date")%></div></td>
        <td width="92" align="left"><div align="center"><%=rs.getString("address")%></div></td>
      </tr>
      <%
               }
            }catch(Exception ex) {System.out.println(ex.getMessage());}
      }
      else if(str.equals("name"))                                            //判断是否根据商品名称进行汇总
       {%>
    </table>

    <table width="596" height="22" border="1">
      <tr bgcolor="#FFCCFF">
        <td width="264"><div align="center"><strong>商品名称</strong></div></td>
        <td width="316"><div align="center"><strong>库存汇总</strong></div></td>
      </tr>
    </table>
    <table width="595" height="22" border="1">
      <%
         try{
            Connection connection=con.getCon();                              //得到一个数据库连接
            Statement st=connection.createStatement();
            sql ="select name,sum(total) sum from tb_clothes group by name order by sum asc";   //利用 SQL 语句进行查询
            ResultSet rs=st.executeQuery(sql);                               //执行 SQL 语句
             while(rs.next()){
            %>
      <tr>
        <td width="265" align="left"><div align="center"><%=rs.getString("name")%></div></td>
        <td width="314" align="left"><div align="center"><%=rs.getString("sum")%>件</div></td>
      </tr>
      <%
              }
            }catch(Exception ex) {System.out.println(ex.getMessage());}
       }
       else                                                                  //根据商品进货地点进行汇总
       {
       %>
    </table>
    <table width="596" height="22" border="1">
      <tr bgcolor="#FFCCFF">
        <td width="266"><div align="center"><strong>进货地点</strong></div></td>
        <td width="317"><div align="center"><strong>库存汇总</strong></div></td>
      </tr>
    </table>
    <table width="596" height="22" border="1">
      <%
       try{
```

```
            Connection connection=con.getCon();                                          //得到一个数据库连接
            Statement st=connection.createStatement();
            sql ="select address,sum(total) sum from tb_clothes group by address order by sum asc";    //利用 SQL 语句进行查询
            ResultSet rs=st.executeQuery(sql);                                           //执行 SQL 语句
             while(rs.next()){
             %>
    <tr>
      <td width="265" align="left"><div align="center"><%=rs.getString("address")%></div></td>
      <td width="317" align="left"><div align="center"><%=rs.getString("sum")%>件</div></td>
    </tr>
    <%
              }
         }catch(Exception ex){System.out.println(ex.getMessage());}
    }
    %>
```

秘笈心法

心法领悟 096：SUM 函数中的数据类型。

SUM 函数只能用于数据类型是 int、smallint、tinyint、decimal、numeric、float、real、money 和 smallmoney 的字段。

实例 097 查询 SQL Server 数据表中的前 3 条数据

光盘位置：光盘\MR\04\097

中级

实用指数：★★☆

实例说明

本实例实现的是在客户信息表中查询前 3 名客户的信息。运行本实例，即可将客户信息表中前 3 名客户的信息显示在下面的表格中，如图 4.11 所示。

查询前三名数据>>>

查询某公司前三名客户信息

姓名	性别	年龄	所在部门	联系电话	地址
章六	男	22	JavaWeb	09893	长春市
马援	女	23	C语言	78965	吉林市
冉小雪	女	19	Asp.net	78678	通化市

图 4.11　查询客户信息表中前 3 条数据

关键技术

要实现查询客户信息表中的前 3 条数据，应该采用 TOP n 返回满足 WHERE 子句的前 n 条记录。其语法如下：

```
SELECT TOP n [PERCENT]
FROM table
WHERE …
ORDER BY…
```

参数说明

[PERCENT]：返回行的百分之 n，而不是 n 行。

如果 SELECT 语句中没有 ORDER BY 子句，TOP n 返回满足 WHERE 子句的前 n 条记录；如果子句中满足条件的记录少于 n，那么仅返回这些记录。

设计过程

（1）创建数据库连接类 DBbean，主要是通过加载驱动程序、建立数据库连接的步骤得到 Connection 类的一个对象。

（2）在 JSP 页面中通过 JSP 的 useBean 标记获取 DBbean 类中的连接，然后通过执行 SQL 语句获得查询结果，并将查询结果输出到表格中。具体代码如下：

```
<%
      try{
       String sql="select top 3 * from tb_client";
       Connection connection=con.getCon();                              //得到一个数据库连接
       Statement st=connection.createStatement();
       ResultSet rs=st.executeQuery(sql);                               //执行 SQL 语句
```

```
            if(rs.next())
            {
            do{                                                    //显示查询结果
%>
        <tr>
            <td width="68" align="left"><div align="center"><%=rs.getString("name")%></div></td>
            <td width="68" align="left"><div align="center"><%=rs.getString("sex")%></div></td>
            <td width="61" align="left"><div align="center"><%=rs.getString("age")%></div></td>
            <td width="75" align="left"><div align="center"><%=rs.getString("sdept")%></div></td>
            <td width="89" align="left"><div align="center"><%=rs.getString("phone")%></div></td>
            <td width="60" align="left"><div align="center"><%=rs.getString("address")%></div></td>
        </tr>
        <%
    }while(rs.next());
    }
    else
    {
        out.println("<br><strong><center>没有你要检索的数据!</center><strong>");
    }
  } catch(Exception ex) {System.out.println(ex.getMessage());}%>
```

秘笈心法

心法领悟 097：TOP n 的位置和 PERCENT 参数的使用。

Top n 的位置一定要在所查询字段的前面，若有 PERCENT 参数，则其要紧随在 n 之后，同时还要注意 PERCENT 参数是返回行的百分之 n，而不是 n 行。

实例 098　查询 SQL Server 数据表中的后 3 条数据

光盘位置：光盘\MR\04\098

中级　实用指数：★★☆

实例说明

本实例实现的是在客户信息表中查询后 3 名客户的信息。运行本实例，即可将客户信息表中后 3 名客户的信息显示在下面的表格中，如图 4.12 所示。

查询后三名数据>>>

查询某公司后三名客户信息

姓名	性别	年龄	所在部门	联系电话	地址
杨康	男	27	Oracle	45673	九台市
秦阳	男	23	PHP	67543	四平市
刘日其	男	26	VB	12567	松原市

图 4.12　查询客户信息表中后 3 条数据

关键技术

实现在 SQL Server 数据库中查询后 3 条记录的方法与查询前 3 条记录的方法基本一致，同样采用 TOP 关键字。但需要注意的是，TOP 关键字本身并没有获取某表中指定数据的能力，要完成获取前几条或后几条数据，需要借助 ORDER BY 子句，先将数据进行排序后再使用 TOP 关键字来限制 SQL 语句返回的行数。

（1）创建数据库连接类 DBbean，主要是通过加载驱动程序、建立数据库连接的步骤得到 Connection 类的一个对象。

（2）在 JSP 页面中通过 JSP 的 useBean 标记获取 DBbean 类中的连接，然后通过执行 SQL 语句获得查询结果，并将查询结果输出到表格中。具体代码如下：

```
<%
try{
        String sql="select top 3 * from tb_client order by id desc";
        Connection connection=con.getCon();                         //得到一个数据库连接
        Statement st=connection.createStatement();
        ResultSet rs=st.executeQuery(sql);                          //执行 SQL 语句
        if(rs.next())
          {
```

```
            do{                                                          //显示查询结果
%>
        <tr>
            <td width="68" align="left"><div align="center"><%=rs.getString("name")%></div></td>
            <td width="68" align="left"><div align="center"><%=rs.getString("sex")%></div></td>
            <td width="61" align="left"><div align="center"><%=rs.getString("age")%></div></td>
            <td width="75" align="left"><div align="center"><%=rs.getString("sdept")%></div></td>
            <td width="89" align="left"><div align="center"><%=rs.getString("phone")%></div></td>
            <td width="60" align="left"><div align="center"><%=rs.getString("address")%></div></td>
        </tr>
        <%
    }while(rs.next());
    }
    else
    {
        out.println("<br><strong><center>没有你要检索的数据!</center><strong>");
    }
  } catch(Exception ex) {System.out.println(ex.getMessage());}%>
```

秘笈心法

心法领悟 098：字段排序默认情况。

在查询记录之前要对相应的字段进行排序，若在字段后面不写关键字 ASC 或者 DESC，则默认采用 ASC 方式进行排序。

实例 099 查询 MySQL 数据表中的前 3 条数据

光盘位置：光盘\MR\04\099

中级

实用指数：★★☆

实例说明

本实例实现的是在学生信息表中查询前 3 名学生的信息。运行本实例，即可将学生信息表中前 3 名学生的信息显示在下面的表格中，如图 4.13 所示。

查询前三名数据>>>

查询班级中前三名学生信息

姓名	年龄	性别	所在班级	所在系别	家庭住址
张远	21	男	20802班	计算机应用	长春
李林	20	女	20801班	图形图像	长春
杨志	22	男	20806班	计算机网络	吉林

图 4.13　查询学生信息表中前 3 条数据

关键技术

MySQL 数据库中提供了 LIMIT 子句来限制 SELECT 语句返回的行数。如果查询数据的 SQL 语句中包含 GROUP BY 与 ORDER BY 子句，则将 LIMIT 子句放在 GROUP BY 子句与 ORDER BY 子句的后面。语法如下：

```
SELECT [DISTIN|UNIQUE](*,columname[AS alias],…)
FROM table
WHERE …
ORDER BY…
LIMIT([offset],rows)
```

参数说明

❶ offset：指定要返回的第一行的偏移量。开始行的偏移量是 0。

❷ rows：指定返回行的最大数目。

设计过程

（1）创建数据库连接类 DBbean，主要是通过加载驱动程序、建立数据库连接的步骤得到 Connection 类的一个对象。

（2）在 JSP 页面中通过 JSP 的 useBean 标记获取 DBbean 类中的连接，然后通过执行 SQL 语句获得查询结果，并将查询结果输出到表格中。具体代码如下：

```
<%
        try{
```

```
        String sql="select * from tb_student limit 0,3";
        Connection connection=con.getCon();                    //得到一个数据库连接
        Statement st=connection.createStatement();
        ResultSet rs=st.executeQuery(sql);                     //执行 SQL 语句
        if(rs.next())
         {
         do{                                                   //显示查询结果
%>
      <tr>
        <td width="86" align="left"><div align="center"><%=rs.getString("name")%></div></td>
        <td width="69" align="left"><div align="center"><%=rs.getString("age")%></div></td>
        <td width="64" align="left"><div align="center"><%=rs.getString("sex")%></div></td>
        <td width="65" align="left"><div align="center"><%=rs.getString("grade")%></div></td>
        <td width="103" align="left"><div align="center"><%=rs.getString("sdept")%></div></td>
        <td width="76" align="left"><div align="center"><%=rs.getString("address")%></div></td>
      </tr>
      <%
     }while(rs.next());
     }
     else
     {
        out.println("<br><strong><center>没有你要检索的数据!</center><strong>");
     }
   } catch(Exception ex) {System.out.println(ex.getMessage());}%>
```

秘笈心法

心法领悟 099：LIMIT 子句的参数。

该子句的第一个参数为可选。如果只给定一个参数，则代表偏移量为 0 的返回行的最大数目。例如，表达式 limit(3)与表达式 limit(0,3)是等价的。

实例 100　查询 MySQL 数据表中的后 3 条数据

光盘位置：光盘\MR\04\100

中级

实用指数：★★★

实例说明

本实例实现的是在学生信息表中查询后 3 名学生的信息。运行实例，即可将学生信息表中后 3 名学生的信息显示在下面的表格中，如图 4.14 所示。

查询后三名数据>>>

查询班级中后三名学生信息

姓名	年龄	性别	所在班级	所在系别	家庭住址
马良	22	男	20804班	电子商务	四平
徐毅	23	男	20803班	计算机应用	白城
李丽	18	女	20804班	电子商务	九台

图 4.14　查询学生信息表中后 3 条数据

关键技术

在 MySQL 数据库中查询后 3 条记录的方法与查询前 3 条记录的方法基本一致，同样采用 LIMIT 关键字。但需要注意的是，LIMIT 关键字本身并没有获取某表中指定数据的能力，要完成获取前几条或后几条数据，需要借助 ORDER BY 子句先将数据进行排序后，再使用 LIMIT 关键字来限制 SQL 语句返回的行数。

（1）创建数据库连接类 DBbean，主要是通过加载驱动程序、建立数据库连接的步骤得到 Connection 类的一个对象。

（2）在 JSP 页面中通过 JSP 的 useBean 标记获取 DBbean 类中的连接，然后通过执行 SQL 语句获得查询结果，并将查询结果输出到表格中，具体代码如下：

```
<%
        try{
          String sql="select * from tb_student order by id desc limit 0,3";
          Connection connection=con.getCon();                              //得到一个数据库连接
          Statement st=connection.createStatement();
          ResultSet rs=st.executeQuery(sql);                               //执行 SQL 语句
          if(rs.next())
            {
            do{                                                            //显示查询结果
%>
        <tr>
          <td width="86" align="left"><div align="center"><%=rs.getString("name")%></div></td>
          <td width="69" align="left"><div align="center"><%=rs.getString("age")%></div></td>
          <td width="64" align="left"><div align="center"><%=rs.getString("sex")%></div></td>
          <td width="65" align="left"><div align="center"><%=rs.getString("grade")%></div></td>
          <td width="103" align="left"><div align="center"><%=rs.getString("sdept")%></div></td>
          <td width="76" align="left"><div align="center"><%=rs.getString("address")%></div></td>
        </tr>
        <%
      }while(rs.next());
      }
      else
      {
          out.println("<br><strong><center>没有你要检索的数据!</center><strong>");
      }
   } catch(Exception ex) {System.out.println(ex.getMessage());}%>
```

秘笈心法

心法领悟 100：LIMIT 子句的位置。

如果查询数据的 SQL 语句中包含 GROUP BY 与 ORDER BY 子句，则 LIMIT 子句一定要写在 GROUP BY 子句与 ORDER BY 子句的后面。

4.2　排序与分组函数的应用

实例 101　按照字母顺序对留学生表进行排序

光盘位置：光盘\MR\04\101

中级

实用指数：★★★

实例说明

在实际应用中，经常会遇到将数据表按某一字母排序的情况。排序分为两种情况，即按字母的升序排序（将字母靠前的内容显示在查询结果的前面）和按字母的降序排序。本实例应用 ORDER BY 子句和 substring()函数，实现将留学生表 tb_abroad 按“名字”字段中的第一个字母进行升序排序，运行结果如图 4.15 所示。

显示查询结果>>>

按字母顺序降序排序　　按字母顺序升序排序

编号	名字	姓氏	国籍
2	armando	PURVIS	加拿大
4	apple	SMITH	英国
3	chuch	TAYLOR	美国
1	kelly	KONG	美国
5	laura	CHRISINT	英国

图 4.15　本实例的运行结果

本实例是按 substring()函数返回的字符串进行升序排序。从图 4.15 中可以看出，2 号留学生由于名字的第 1 个字母为“a”，所以排在了第 1 位。

substring()函数主要用于返回字符、binary、text 或 image 表达式的一部分。

其语法如下：

```
substring(expression,start,length)
```

参数说明

❶ expression：字符串、二进制字符串、text、image、列或包含列的表达式。不可以使用包含聚合函数的表达式。

❷ start：一个整数，用于指定字符串的开始位置。

❸ length：一个整数，用于指定字符串的长度（要返回的字符数或字节数）。

设计过程

（1）在项目中创建类 AbroadUtil，在该类中定义操作数据库的各种方法，然后定义将留学生表进行升序或者降序排序的方法。具体代码如下：

```
public List getList(String order){
	List list = new ArrayList();                                    //定义保存查询结果的 List 集合
	conn = getConnection();
	try {
		Statement statmenet = conn.createStatement();               //获取 Statement 对象
		String sql = "select * from tb_abroad order by substring(name,1,1) "+order;   //定义排序查询的 SQL 语句
		ResultSet rest = statmenet.executeQuery(sql);               //执行查询语句，返回查询结果集
		while(rest.next()){                                         //循环遍历查询结果集
			Abroad abord = new Abroad();                            //定义与数据表对应的 Abroad 对象
			abord.setId(rest.getInt(1));                            //应用查询结果设置对象属性
			abord.setName(rest.getString(2));
			abord.setSurname(rest.getString(3));
			abord.setNationality(rest.getString(4));
			list.add(abord);                                        //向集合中添加对象
		}
	} catch (Exception e) {
		e.printStackTrace();
	}
	return list;                                                    //返回查询结果
}
```

（2）在 list.jsp 页面中定义表格，显示查询结果。具体代码如下：

```
<table width="353" height="65" border="1" align="center">
  <tr>
    <td width="66"><div align="center">编号</div></td>
    <td width="93"><div align="center">名字</div></td>
    <td width="67"><div align="center">姓氏</div></td>
    <td width="99"><div align="center">国籍</div></td>
  </tr>
  <%
     List list = (List)request.getAttribute("list");              //获取保存在 request 对象中的对象值
     for(int i = 0;i<list.size();i++){                              //循环遍历集合对象
         Abroad abroad = (Abroad)list.get(i);                       //获取集合中对象
  %>
  <tr>
    <td><div align="center"><%=abroad.getId()%></div></td>          //将对象属性显示在页面中
    <td><div align="center"><%=abroad.getName()%></div></td>
    <td><div align="center"><%=abroad.getSurname()%></div></td>
    <td><div align="center"><%=abroad.getNationality()%></div></td>
  </tr>
  <%} %>
</table>
```

心法领悟 101：Oracle 数据库中的字符串截取函数。

SQL 语句在各数据库之间是通用的，但有些函数并不是通用的，例如 SQL Server 数据库中的 TOP 关键字就不适用于其他数据库。同样，本实例应用的 substring()函数可以应用到 SQLServer 和 MySQL 数据库中，但是不能应用在 Oracle 数据库中。Oracle 数据库中的字符串截取函数是 SUBSTR。若想按数据表（tb_abstr06）“名字”字段中第 1 个字母排序，在 Oracle 数据库中可写成如下 SQL 语句。

```
select * from tb_abstu06 order by substr(名字,1,1)
```

实例 102　按姓氏笔画排序

光盘位置：光盘\MR\04\102

高级

实用指数：★★★★

实例说明

按姓氏笔画排序是一种很常见的排序方式，例如，图书中很多作者排名都是按照姓氏笔画排序的。SQL Server 数据库中提供了 COLLATE 子句，可实现将数据表中数据按姓氏笔画排序。本实例实现的是将员工表中数据按姓氏笔画排序后在页面中进行显示，如图 4.16 所示。

查询条件信息>>>

按照姓氏笔画排序

编号	姓名	职务	入司时间
1	王丽	部门经理	2007/10/7
3	刘静	Java程序员	2004/12/5
4	陈双	质量检测	2008/10/5
2	葛蕾	副总经理	2005/5/6
5	蓝蓝	文职	2010/11/5

图 4.16　按照姓氏笔画排序

关键技术

COLLATE 子句可应用于数据库定义或列定义以定义排序规则，或应用于字符串表达式以应用排序规则投影。其语法如下：

```
COLLATE < collation_name >< collation_name > ::= { Windows_collation_name } | { SQL_collation_name }
```

参数说明

❶ collation_name：应用于表达式、列定义或数据库定义的排序规则的名称。

❷ Windows_collation_name：Windows 排序规则名称。

❸ SQL_collation_name：SQL 排序规则名称。

COLLATE 子句只能应用于 char、varchar、text、nchar、nvarchar 和 ntext 数据类型。排序规则一般由排序规则名标识，例如 chinese_prc_stroke_cs_as_ks_ws 表示的就是排序规则。其中，chinese_prc_是指汉字的 UNICODE 排序规则；stroke 标识查询结果按姓氏笔画排序。排序规则的后半部分后缀含义分别介绍如下。

- _bin：二进制排序。
- _ci(cs)：是否区分大小写，ci 表示不区分，cs 表示区分。如果要将比较大写字母和小写字母视为不等，可以选择该选项。
- _ai(as)：是否区分重音，ai 表示不区分，as 表示区分。如果要将比较重音和非重音字母视为不等，可以选择该选项。如果选择该选项，比较还将重音不同的字母视为不等。
- _ki(ks)：是否区分假名类型，ki 表示不区分，ks 表示区分。如果要将比较片假名和平假名日语音节视为不等，可以选择该选项。
- _wi(ws)：是否区分宽度，wi 表示不区分，ws 表示区分。如果要将比较半角字符和全角字符视为不等，可以选择该选项。

（1）在项目中创建类 PersonnelUtil，在该类中定义查询数据方法 getPersonnel()，该方法以 List 集合返回查询结果集。具体代码如下：

```
public List getPersonnel() {
    List list = new ArrayList();                                                    //定义用于保存返回值的 List 集合
    conn = getConn();                                                               //获取数据库连接
    try {
        Statement staement = conn.createStatement();
        String sql = "select * from tb_personnel order by name   collate chinese_prc_stroke_cs_as_ks_ws";   //定义按笔画排序的 SQL 语句
        ResultSet set = staement.executeQuery(sql);                                 //执行查询语句，返回查询结果集
        while (set.next()) {                                                        //循环遍历查询结果集
            Personnel personnel = new Personnel();                                  //创建与数据表对应的 JavaBean 对象
            personnel.setId(set.getInt(1));                                         //应用查询结果设置对象属性
            personnel.setName(set.getString(2));
            personnel.setHeadship(set.getString(3));
            personnel.setDdate(set.getString(4));
            list.add(personnel);                                                    //向集合中添加对象
```

```
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return list;                                                          //返回查询集合
}
```

（2）在 list.jsp 页面中，定义表格显示查询结果。具体代码如下：

```
<table width="399" height="71" border="1" align="center">
  <tr>
    <td><div align="center">编号</div></td>
    <td><div align="center">姓名</div></td>
    <td><div align="center">职务</div></td>
    <td><div align="center">入司时间</div></td>
  </tr>
  <%if(request.getAttribute("list") != null){                     //判断保存在 request 对象中的对象是否为空
      List list = (List)request.getAttribute("list");             //获取保存在 request 对象中的集合
      for(int i = 0;i<list.size();i++){                           //循环遍历 List 集合
            Personnel personn = (Personnel)list.get(i);           //获取 List 集合中的对象
  %>
  <tr>
    <td><div align="center"><%=personn.getId()%></div></td>              //在页面中显示对象属性
    <td><div align="center"><%=personn.getName()%></div></td>
    <td><div align="center"><%=personn.getHeadship()%></div></td>
    <td><div align="center"><%=personn.getDdate()%></div></td>
  </tr>
  <%} }%>
</table>
```

秘笈心法

心法领悟 102：获取 SQL Server 支持的排序规则。

在查询分析器中执行如下语句，可以得到 SQL Server 支持的所有排序规则。

```
select * from ::fn_helpcollations()
```

实例 103　将汉字按音序排序

光盘位置：光盘\MR\04\103

高级

实用指数：★★★

实例说明

SQL Server 汉字排序规则可以按音序、笔画排序。本实例实现的是将员工表中数据按姓名音序进行排序后显示在页面中，如图 4.17 所示。

查询条件信息>>>

按照音序进行排序

编号	姓名	职务	入司时间
1	王丽	部门经理	2007/10/7
3	刘静	Java程序员	2004/12/5
4	陈双	质量检测	2006/10/5
2	葛雷	副总经理	2005/5/6
5	蓝蓝	文职	2010/11/5

图 4.17　将汉字按音序排序

关键技术

按音序排序同样可以应用 COLLATE 函数来完成，chinese_prc_ci_as 指定排序规则。其排序规则可以参考按姓氏笔画排序的规则。不使用后缀_stroke，数据库会按指定的表达式音序进行排序。

在项目中创建类 PersonnelUtil，在该类中定义按音序进行数据排序的方法 getPersonnel()，该方法以 List 集合作为返回值。具体代码如下：

```
public List getPersonnel() {
        List list = new ArrayList();                                   //定义用于保存返回值的 List 集合
        conn = getConn();                                              //获取数据库连接
        try {
            Statement staement = conn.createStatement();
```

```
            String sql = "select * from tb_personnel order by name collate chinese_prc_stroke_cs_as";    //定义按音序排序的 SQL 语句
            ResultSet set = staement.executeQuery(sql);                                                  //执行查询语句，返回查询结果集
            while (set.next()) {                                                                         //循环遍历查询结果集
                Personnel personnel = new Personnel();                                                   //创建与数据表对应的 JavaBean 对象
                personnel.setId(set.getInt(1));                                                          //应用查询结果设置对象属性
                personnel.setName(set.getString(2));
                personnel.setHeadship(set.getString(3));
                personnel.setDdate(set.getString(4));
                list.add(personnel);                                                                     //向集合中添加对象
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return list;                                                                                     //返回查询集合
    }
```

秘笈心法

心法领悟 103：不要使用 get 方式提交包含口令的 form 表单。

切记不要使用 get 方式来提交询问口令的表单，否则访问输入的口令将不作为 URL 的一部分，存储在多个地方，其中包含 Web 浏览器的历史记录文件和 Web 服务器日志。

实例 104 按列的编号排序

光盘位置：光盘\MR\04\104

中级

实用指数：★★★

实例说明

ORDER BY 子句支持多种方式的排序，如通过列名称进行排序、通过列的编号进行排序等。本实例使用的是按列的编号对图书表进行排序，运行结果如图 4.18 所示。

按列的编号进行排序>>>

按销量升序排序　　按销量降序排序

编号	图书名称	销量	销售日期
S002	Java从入门到精通	360	2010/12/4
S001	Java程序开发完成自学手册	450	2010/7/8
S003	JavaWeb开发实战宝典	490	2010/11/26
S004	JavaWeb典型模块大全	580	2010/10/12
S005	Jsp程序开发案例全程实录	610	2010/9/6

图 4.18　按列的编号排序

关键技术

在 ORDER BY 子句中，用列编号表示列有两个作用，一是进行缩写，可以减少击键次数；二是当需要排序的列是计算得到的结果时，使用列序号是必需的。所谓列序号指的是该列在 SELECT 子句中的位置，最左边的序号为 1，相邻的下一个位置为 2，以此类推。

注意：按列序号排序时要注意，列的序号指的是 SELECT 子句中的顺序，而不是数据库中表存储的顺序。

在使用列序号进行排序时应注意，如果使用“*”查询数据表中所有列和计算结果列，排序时同样可以使用计算后的结果列进行排序。例如，查询学分表中的全部信息以及总成绩，并按总成绩进行排序，代码如下：

```
select *,(math+english+chinese) as  总成绩  from tb_score02
order by   2
```

在项目中创建类 BookUtil，在该类中定义按照图书表中的图书销量进行排序的 getBookOrder()。该方法有一个 String 类型的参数，用于指定是按照升序排序还是按降序排序。由于在图书表中图书销量是第 4 列，因此可以使用编号“4”进行排序查询。具体代码如下：

```
public List getBookOrder(String order) {
        List list = new ArrayList();
        conn = getConnection();                                          //调用获取数据库连接方法
        try {
            Statement statement = conn.createStatement();                //获取 Statement 对象
            String sql = "select * from tb_booksell order by 4 "+order;   //定义查询 SQL 语句
```

```
            ResultSet rest = statement.executeQuery(sql);                    //执行查询语句获取查询结果集
            while (rest.next()) {                                            //循环遍历查询结果集
                BookSell bookSell = new BookSell();                          //定义与图书表对应的 JavaBean 对象
                bookSell.setId(rest.getInt(1));                              //应用查询结果设置 JavaBean 参数值
                bookSell.setBookId(rest.getString(2));
                bookSell.setBookName(rest.getString(3));
                bookSell.setTotal(rest.getInt(4));
                bookSell.setSellDate(rest.getString(5));
                list.add(bookSell);                                          //将 JavaBean 添加到集合中
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return list;                                                         //返回查询结果
}
```

秘笈心法

心法领悟 104：标识符的种类。

数据库对象的名称被看成是该对象的标识符。Microsoft® SQL Server™ 中的每一项内容都可带有标识符，如服务器、数据库和数据库（例如，表、视图、列、索引、触发器、过程、约束、规则等）都有标识符。大多数对象要求带有标识符，但对某些对象（如约束）来说标识符是可选项。标识符有两大类，分别为常规标识符和分隔标识符。常规标识符的格式规则是，在 Transact-SQL 语句中使用常规标识符时不用将其分隔；分隔标识符包含在双引号（"）或者方括号（[ ]）内。符合标识符格式规则的标识符可以分隔，也可以不分隔。

实例 105　从表中随机返回记录

光盘位置：光盘\MR\04\105

中级　实用指数：★★☆

实例说明

SQL Server 数据库中提供了两个随机数函数，分别为 RAND 函数与 NOWID 函数。其中 RAND 函数在一个查询中只能返回一个结果，NOWID 函数返回的列可以进行 ORDER BY 排序处理。本实例使用 NOWID 函数实现从表中随机返回 3 条记录，如图 4.19 所示。在如图 4.19 所示页面中单击“查询”按钮时，每次的显示结果都是不同的。

查询条件信息>>>

随机获取员工表中的3条数据： 查询

编号	姓名	职务	入司时间
2	葛雷	副总经理	2005/5/6
4	陈双	质量检测	2006/10/5
1	王丽	部门经理	2007/10/7

图 4.19　随机获取员工表中的 3 条数据

要从数据表中随机返回数据记录信息或者对数据记录进行随机排序，可以使用随机数函数。由于 RAND 函数在一个查询中只能返回一个结果，因此可以在 NOWID 函数返回的列上进行 ORDER BY 分组处理。

注意：上述方法在查询某个表的所有数据时，会要求对整个表进行扫描，然后产生一个计算列再进行排序。建议最好不要对较大的表进行这样的操作，因为速度会很慢。

设计过程

在项目中创建类 PersonnelUtil，在该类中定义操作数据库的各种方法，然后定义随机获取数据表中的 3 条数据的方法 getPersonnel()，并将查询结果以 List 集合的形式返回。具体代码如下：

```
public List getPersonnel() {
        List list = new ArrayList();                                         //定义用于保存返回值的 List 集合
        conn = getConn();                                                    //获取数据库连接
        try {
            Statement staement = conn.createStatement();
            String sql = "select top 3 * from tb_personnel    order by newid()";   //定义随机获取 3 条数据的方法
            ResultSet set = staement.executeQuery(sql);                      //执行查询语句返回查询结果集
```

```
        while (set.next()) {                                    //循环遍历查询结果集
            Personnel personnel = new Personnel();              //创建与数据表对应的 JavaBean 对象
            personnel.setId(set.getInt(1));                     //应用查询结果设置对象属性
            personnel.setName(set.getString(2));
            personnel.setHeadship(set.getString(3));
            personnel.setDdate(set.getString(4));
            list.add(personnel);                                //向集合中添加对象
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return list;                                                //返回查询集合
}
```

秘笈心法

心法领悟 105：查找网络服务程序监听异常的问题。

如果 Tomcat 服务程序处于启动状态，但浏览器提示“该页无法显示”或者返回的不是 Tomcat 提供的首页，很可能是因为 Tomcat 服务程序所使用的网络监听端口号已被其他网络服务程序或 Web 服务程序占用，导致 Tomcat 服务程序并没有真正正常启动运行。在命令行窗口中执行 netstat -na 命令，可查看 TCP 监听端口列表中是否包含 Tomcat 的 Web 服务绑定的监听端口。

实例 106　使用 GROUP BY 子句实现数据的分组统计

光盘位置：光盘\MR\04\106

中级　实用指数：★★★

实例说明

在实际开发中，对数据的分组查询是很重要的一种形式。使用 GROUP BY 子句可以实现数据的分组统计。本实例实现的是将订单表中的数据进行分组统计，用户可以选择按销售代表名称分组，也可以选择按订单区域分组，运行结果如图 4.20 所示。

对数据进行分组统计>>>

按销售代表名称分组　按订单区域分组

编号	姓名	地址	金额	签订日期
3	李四	山东青岛	85000.0	2010-12-03
2	王五	沈阳大连	286000.0	2010-11-20
4	小双	上海	60000.0	2010-12-05
5	小碗	北京	540000.0	2010-10-12
1	张三	山东淄博	270000.0	2010-12-25

图 4.20　使用 GROUP BY 子句实现数据的分组统计

实现数据分组时，可使用 GROUP BY 子句与聚集函数相结合获取所需要的数据。GROUP BY 子句的作用是指定用来放置输出行的组，其语法如下：

```
[GROUP BY[ALL] expression[,…n]
[WHERE {CUBE|ROLLUP}]
]
```

参数说明

❶ ALL：包含于选定列表中匹配的所有组和结果集。如果用户指定了 ALL，将对组中不满足搜索条件的汇总列返回空值。

❷ expression：对查询执行分组的表达式，即要进行分组的列。在选择列表内定义的列的别名，不能用于指定分组列。

❸ CUBE：指定在查询的结果集内不仅包含 GROUP BY 子句提供的正常行，还包含汇总行。

❹ ROLLUP：指定在结果集内不仅包含 GROUP BY 子句提供的正常行，还包含汇总行。

注意：如果 SELECT 子句中包含聚集函数，同时使用 GROUP BY 进行汇总统计，则 SELECT 语句中列出的非聚集表达式内的所有列都包含在 GROUP BY 列表中，或者 GROUP BY 表达式必须与选择列表表达式完全匹配。

设计过程

（1）在项目中定义 OrderUtil 类，在该类中定义操作数据库的相关方法，然后定义按照指定的列表查询数据库的方法 getGroupBy()，并将查询结果以 List 形式返回。具体代码如下：

```
public List getGroupBy(String row){
        List list = new ArrayList();                                                    //定义保存查询结果的 List 对象
        String sql = "select id,name,address,sum(money) as money,oDate from tb_order group by "+row;    //定义查询 SQL 语句
        conn = getConnection();                                                         //调用创建数据库连接方法
        try {
                Statement statement = conn.createStatement();                           //获取 Statement 对象
                ResultSet rest = statement.executeQuery(sql);                           //执行查询语句获取查询结果集
                while(rest.next()){                                                     //循环遍历查询结果集
                        Order order = new Order();                                      //定义数据表对应的 JavaBean 对象
                        order.setId(rest.getInt(1));
                        order.setName(rest.getString(2));
                        order.setAddress(rest.getString(3));
                        order.setMoney(rest.getFloat(4));
                        order.setoDate(rest.getString(5));
                        list.add(order);                                                //向集合中添加对象
                }
        } catch (SQLException e) {
                e.printStackTrace();
        }
        return list;                                                                    //返回保存查询结果的 List 集合对象
}
```

（2）创建名为 OrderServlet 的 Servlet 类，由该类处理用户提交的请求，并调用 OrderUtil 类中的方法对数据进行分组查询，然后将查询结果保存在 request 对象中。具体代码如下：

```
public void getGroup(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        OrderUtil outil = new OrderUtil();                                              //创建操作数据库的类对象
        List list = new ArrayList();                                                    //定义保存查询结果的集合对象
        String message = request.getParameter("Submit");                                //获取用户提交查询条件
        if(message.equals("按销售代表名称分组")){                                        //判断查询条件
                list = outil.getGroupBy("name");                                        //调用查询方法
        }
        if(message.equals("按订单区域分组")){
                list = outil.getGroupBy("address");
        }
        request.setAttribute("list", list);                                             //将查询结果保存在 request 对象中
        request.getRequestDispatcher("list.jsp").forward(request, response);            //设置请求转发地址
}
```

秘笈心法

心法领悟 106：使用字符过滤器。

在 Web 程序开发中，中文的编码问题经常会遇到，例如，在中文参数传递、查询数据库中。此时就要用到字符过滤器，在每次请求中都进行转码，即可解决中文乱码问题。

实例 107	利用 GROUP BY 子句实现多表分组统计 光盘位置：光盘\MR\04\107	中级 实用指数：★★☆

实例说明

将两张表进行连接后，可以使用 GROUP BY 子句进行分组统计查询。本实例涉及两张表，即部门表与员工

表。部门表中存储的是各部门的名称，员工表中存储的是员工的信息，员工表中的 did 键对应部门表中的 id 键。本实例实现查询这两张表，将各部门的总工资显示在页面中，运行结果如图 4.21 所示。

统计查询>>>

查询各部门的总工资

部门名称	工资
Java开发部	10400
VB开发部	6800
质量部	7800
VC开发部	7900
市场部	12100

图 4.21　分组统计各部门的总工资

关键技术

本实例在进行分组统计查询时，用到了 GROUP BY 子句，在数据统计中用到了 SUM 函数。有关 GROUP BY 子句的语法参见实例 106，SUM 函数的语法参见实例 087，这里不再赘述。

设计过程

（1）在项目中创建类 EmpUtil，在该类中定义操作数据库的方法，然后定义查询部门表与员工表获取每个部门总工资的方法 getEmp()，该方法以 List 形式返回查询结果。具体代码如下：

```
public List getEmp(){
	List list = new ArrayList();						//定义保存查询结果的 List 集合对象
	conn = getConnection();							//获取数据库连接
	try {
		Statement statement = conn.createStatement();			//获取 Statement 对象
		String sql = "select d.dName,sum(laborage) from tb_emp e,tb_dept d where e.did = d.id group by did";	//定义查询数据库的 SQL 语句
		ResultSet rest = statement.executeQuery(sql);			//执行查询语句，获取查询结果集
		while(rest.next()){							//循环遍历查询结果集
			Emp emp = new Emp();					//定义与数据表对应的 JavaBean 对象
			emp.setDept(rest.getString(1));				//应用查询结果设置对象属性
			emp.setLaborage(rest.getString(2));
			list.add(emp);						//向集合中添加对象
		}
	} catch (SQLException e) {
		e.printStackTrace();
	}
	return list;								//返回查询结果集
}
```

（2）创建名为 GetMessageServlet 的 Servlet 类，在该类中定义操作数据库的类对象，调用查询方法，并将查询结果保存到 request 对象中。具体代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
		throws ServletException, IOException {
	EmpUtil empUtil = new EmpUtil();						//创建操作数据库的类对象
	List list = empUtil.getEmp();						//调用查询方法
	request.setAttribute("list",list);					//将查询结果保存在 request 对象中
	request.getRequestDispatcher("list.jsp").forward(request, response);		//将请求转发至 list.jsp 页
}
```

心法领悟 107：如何区分 0、空字符串和 null。

在程序开发中，经常会遇到 0、空字符串和 null。这些看起来都是空的意思，那么它们在使用时有什么区别呢？

在 Java 中，对于声明后未赋值的数值类型变量，其默认值为 0；对于声明后未赋值的字符串变量，则默认值为空字符串；null 说明变量不包含有效数据，它是将 null 值显式地赋值给变量的结果，也可能是包含 null 的表达式之间进行运算的结果。

4.3　比较大小与逻辑应用

实例 108　在查询结果中不显示重复记录

光盘位置：光盘\MR\04\108

中级

实用指数：★★☆

实例说明

本实例实现的是在水果信息表中查询不重复的水果信息。运行本实例，单击“查询不重复记录”按钮，即可将水果信息表中的水果信息显示在下面的表格中，同时把重复的记录删除，如图 4.22 所示。

查询不重复记录>>>

查询已销售水果信息　查询不重复记录

名称	价格	生产日期	产地	质里
苹果	2	2010-09-08	河南	优
香蕉	3	2010-09-08	山东	良
橘子	1	2010-09-07	云南	优
菠萝	4	2010-08-02	海南	良
荔枝	8	2010-09-01	福建	良
甘蔗	3	2010-07-04	山西	良

图 4.22　在查询结果中不显示重复记录

关键技术

查询时不显示重复记录主要应用了 DISTINCT 关键字，该关键字用于删除重复记录。

在实现查询操作时，如果查询的选择列表中包含一个表的主键，那么每个查询结果中的记录都将是唯一的（因为主键在每一条记录中有一个不同的值）；如果主键不包含在查询结果中，就可能出现重复记录。使用 DISTINCT 关键字以后即可删除重复记录。

DISTINCT 的语法如下：

```
SELECT DISTINCT select_list
```

注意：DISTINCT 关键字并不是指某一行，而是指不重复 SELECT 输出的所有列。这一点十分重要，其作用是防止相同的行出现在一个查询结果的输出中。

（1）创建数据库连接类 DBbean，主要是通过加载驱动程序、建立数据库连接的步骤得到 Connection 类的一个对象。

（2）在 JSP 页面中通过 JSP 的 useBean 标记获取 DBbean 类中的连接，然后通过执行 SQL 语句获得查询结果，并将查询结果输出到表格中。具体代码如下：

```
<%
          String sql="";
          String order=request.getParameter("order");
     try{
      if(order==null)
      {
       sql="select name,price,date,address,quality from tb_fruitsell";
      }
      else
      {
       sql="select distinct name,price,date,address,quality from tb_fruitsell";
      }
      Connection connection=con.getCon();                          //得到一个数据库连接
      Statement st=connection.createStatement();
      ResultSet rs=st.executeQuery(sql);                           //执行 SQL 语句
      if(rs.next())
       {
       do{                                                         //显示查询结果
%>
     <tr>
       <td width="104" align="left"><div align="center"><%=rs.getString("name")%></div></td>
       <td width="78" align="left"><div align="center"><%=rs.getString("price")%></div></td>
```

```
            <td width="116" align="left"><div align="center"><%=rs.getString("date")%></div></td>
            <td width="94" align="left"><div align="center"><%=rs.getString("address")%></div></td>
            <td width="106" align="left"><div align="center"><%=rs.getString("quality")%></div></td>
          </tr>
          <%
      }while(rs.next());
      }
      else
      {
          out.println("<br><strong><center>没有你要检索的数据!</center><strong>");
      }
  } catch(Exception ex) {System.out.println(ex.getMessage());}%>
```

秘笈心法

心法领悟 108：设置文本框隐藏属性。

单击“查询不重复记录”按钮，即可在表格中显示不重复记录的数据。其核心技术是设置了文本框隐藏属性，通过判断文本框中数据是否为空来写相应的 SQL 语句。文本框隐藏有两种方式：第一种是设置 input 标记中 style 属性值为 display:none；第二种方式是设置 style 属性值为 visibility:hidden。

实例 109　使用 NOT 查询不满足条件的记录

光盘位置：光盘\MR\04\109

中级

实用指数：★★☆

实例说明

本实例实现的是在图书销售表中查询销售日期不在 2010-06-01—2010-08-01 之间的图书信息。运行实例，单击“查询不满足条件的记录”按钮，即可将图书销售表中销售日期不在 2010-06-01—2010-08-01 之间的图书信息显示在下面的表格中，如图 4.23 所示。

查询不满足条件的记录>>>

日期不在2010-06-01—2010-08-01之间的图书信息　[查询不满足条件的记录]

图书编号	图书名称	图书作者	图书价格	销售日期
4	ASP.NET 从入门到精通（第2版）	庞娅娟	79元	2010-08-09
5	PHP 从入门到精通（第2版）	潘凯华	69元	2010-10-01
6	JSP项目开发案例全程实录（第2版）	明日科技	67元	2010-10-28
7	C#项目开发案例全程实录（第2版）	明日科技	68元	2010-11-09
8	Java 项目开发案例全程实录（第2版）	明日科技	69元	2010-11-28
9	PHP 项目开发案例全程实录（第2版）	明日科技	69元	2010-12-05

图 4.23　查询不满足条件的记录

本实例使用 NOT 与谓词进行组合所形成的条件进行查询。

NOT 与谓词进行组合所形成的表达式分别是[NOT] BETWEEN、IS [NOT] NULL 和[NOT] IN。

（1）[NOT] BETWEEN

该条件指定值的包含范围，使用 AND 将开始值和结束值分开。

其语法如下：

```
test_expression [NOT] BETWEEN begin_expression AND end_expression
```

结果类型为 Boolean，返回的值为：如果 test_expression 的值小于等于 begin_expression 的值或者大于等于 end_expression 的值，则 NOT BETWEEN 返回 true。

注意：若要指定排除范围，还可以使用大于（>）和小于（<）运算符来代替 BETWEEN。

（2）IS [NOT] NULL

根据所使用的关键字指定对空值或非空值进行查询，如果有任何操作数是 null，表达式取值为 null。

（3）[NOT] IN

根据所使用的关键字是包含在列表内还是排除在列表外，指定对表达式进行查询。查询表达式可以使用常

量或列名，而列表可以是一组常量或子查询（更多情况下）。如果列表为一组常量，则应该放置在一对圆括号内。

其语法如下：

```
test_expression[NOT] IN
(
 subquery
 expression[,…n]
)
```

参数说明

❶ test_expression：SQL 表达式。

❷ subquery：包含某列结果集的子查询，该列必须与 test_expression 具有相同的数据类型。

❸ expression[,…n]：一个表达式列表，用来测试是否匹配。所有的表达式必须和 test_expression 具有相同的数据类型。

该语句的返回值是把 test_expression 与 subquery 返回的值进行比较，如果两个值相等，或与逗号分隔的列表中的任何 expression 相等，那么结果值为 true，否则结果值为 false。

设计过程

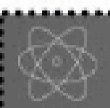
（1）创建数据库连接类 DBbean，主要是通过加载驱动程序、建立数据库连接的步骤得到 Connection 类的一个对象。

（2）在 JSP 页面中通过 JSP 的 useBean 标记获取 DBbean 类中的连接，然后通过执行 SQL 语句获得查询结果，并将查询结果输出到表格中。具体代码如下：

```
<%
          String sql="";
          String order=request.getParameter("order");
      try{
       if(order==null)
       {
        sql="select * from tb_booksell";
       }
       else
       {
        sql="select * from tb_booksell where selldate not between'2010-06-01' and '2010-08-01'";
       }
       Connection connection=con.getCon();                              //得到一个数据库连接
       Statement st=connection.createStatement();
       ResultSet rs=st.executeQuery(sql);                               //执行 SQL 语句
       if(rs.next())
        {
        do{                                                             //显示查询结果
%>
      <tr>
        <td width="59" align="left"><div align="center"><%=rs.getString("id")%></div></td>
        <td width="227" align="left"><div align="center"><%=rs.getString("bookname")%></div></td>
        <td width="75" align="left"><div align="center"><%=rs.getString("author")%></div></td>
        <td width="79" align="left"><div align="center"><%=rs.getString("price")%>元</div></td>
        <td width="139" align="left"><div align="center"><%=rs.getString("selldate")%></div></td>
      </tr>
      <%
    }while(rs.next());
    }
    else
    {
       out.println("<br><strong><center>没有你要检索的数据!</center><strong>");
    }
  } catch(Exception ex) {System.out.println(ex.getMessage());}%>
```

心法领悟 109：SQL 语句中常量的使用。

如果查询数据的 SQL 语句中包含常量，则一定要用单引号（''）把常量括起来。



实例 110 使用 BETWEEN 进行区间查询

光盘位置：光盘\MR\04\110

中级

实用指数：★★☆

实例说明

本实例实现的是在学生信息表中查询年龄在 20～23 周岁之间的学生信息。运行本实例，单击“区间查询”按钮，即可将学生信息表中年龄在 20～23 周岁之间的学生信息显示在下面的表格中，如图 4.24 所示。

区间查询>>>

查询年龄在20-23周岁之间的学生信息 区间查询

学生编号	姓名	性别	年龄	所在班级
1	张远	男	21	20802班
2	李林	女	20	20801班
3	杨志	男	22	20806班
5	刘云	女	20	20805班
8	徐毅	男	23	20803班
9	马良	男	22	20804班

图 4.24　查询指定年龄段的学生信息

关键技术

要实现对指定年龄段的数据查询，需在 SQL 语句中使用 BETWEEN 运算符。

BETWEEN 运算符可以用在 WHERE 子句中选取给定范围的列值所在行。

语法：

```
test_expression [ NOT ] BETWEEN begin_expression AND end_expression
```

参数说明

❶ test_expression：用于在由 begin_expression 和 end_expression 定义的范围内进行测试的表达式。test_expression 必须与 begin_expression 和 end_expression 具有相同的数据类型。

❷ NOT：指定谓词的结果被取反。

❸ begin_expression：任何有效的 SQL 表达式。begin_expression 必须与 test_expression 和 end_expression 具有相同的数据类型。

❹ end_expression：任何有效的 SQL 表达式。end_expression 必须与 test_expression 和 begin_expression 具有相同的数据类型。

❺ AND：作为一个占位符，表示 test_expression 应该处于由 begin_expression 和 end_expression 指定的范围内。

（1）创建数据库连接类 DBbean，主要是通过加载驱动程序、建立数据库连接的步骤得到 Connection 类的一个对象。

（2）在 JSP 页面中通过 JSP 的 useBean 标记获取 DBbean 类中的连接，然后通过执行 SQL 语句获得查询结果，并将查询结果输出到表格中。具体代码如下：

```
<%
            String sql="";
            String order=request.getParameter("order");
        try{
         if(order==null)
         {
          sql="select * from tb_student";
         }
         else
         {
          sql="select * from tb_student where age between '20' and '23'";
         }
         Connection connection=con.getCon();                                    //得到一个数据库连接
         Statement st=connection.createStatement();
         ResultSet rs=st.executeQuery(sql);                                     //执行 SQL 语句
         if(rs.next())
```

```
            {
            do{                                                                              //显示查询结果
%>
        <tr>
            <td width="84" align="left"><div align="center"><%=rs.getString("id")%></div></td>
            <td width="109" align="left"><div align="center"><%=rs.getString("name")%></div></td>
            <td width="98" align="left"><div align="center"><%=rs.getString("sex")%></div></td>
            <td width="80" align="left"><div align="center"><%=rs.getString("age")%></div></td>
            <td width="107" align="left"><div align="center"><%=rs.getString("grade")%></div></td>
        </tr>
        <%
    }while(rs.next());
    }
    else
    {
        out.println("<br><strong><center>没有你要检索的数据!</center><strong>");
    }
  } catch(Exception ex) {System.out.println(ex.getMessage());}%>
```

秘笈心法

心法领悟 110：BETWEEN 运算符的作用范围。

BETWEEN 运算符是包含性的，即首尾的值也是符合条件的。

实例 111　使用关系运算符查询某一时间段的数据

光盘位置：光盘\MR\04\111

中级

实用指数：★★☆

实例说明

本实例利用关系运算符查询指定时间段的图书销售情况。运行程序，在文本框中输入要查询的时间段，如 2010-08-01—2010-11-09，单击“查询”按钮，即可将 2010-08-01—2010-11-09 的图书销售情况显示在表格中，如图 4.25 所示。

开始日期：　截止日期：　查询

书号	书名	作者	价钱	出售日期
4	ASP.NET 从入门到精通（第2版）	庞娅娟	79	2010-08-09
5	PHP 从入门到精通（第2版）	潘凯华	69	2010-10-01
6	JSP项目开发案例全程实录（第2版）	明日科技	67	2010-10-28
7	C#项目开发案例全程实录（第2版）	明日科技	68	2010-11-09

图 4.25　使用关系运算符查询某一时间段的数据

关键技术

要实现对指定时间段数据的查询，需在 SQL 语句中使用“>=”和“<=”关系运算符。关系运算符可以用在 WHERE 子句中选取给定范围的列值所在行，其语法如下：

```
expression>=expression1 and expresstion<=expression2
```

参数说明

❶ expression：用于在由 expression1 和 expression2 定义的范围内进行测试的表达式。expression 必须与 expression1 和 expression2 具有相同的数据类型。

❷ expression1：有效的 SQL 表达式，与 expression 和 expression2 具有相同的数据类型。

如果 expression 的值大于或等于 expression1 的值并且小于或等于 expression2 的值，则结果为 true；如果 expression 的值小于 expression1 的值或者大于 expression2 的值，则结果为 false。

下面举例说明“>=”和“<=”运算符的用法。

（1）使用关系运算符查询年度销售额在 200～500 之间（包括 200 和 500）的图书名称，代码如下：

```
select title,sales from titles where sales>=200 and sales<=500
```

（2）使用关系运算符查询销售日期在 2008-09-10—2010-09-08 时间段的图书信息。

```
select bookname,author,price,publisher from booksell where date>='2008-09-10'and date<='2010-09-08'
```

（1）创建数据库连接类 DBbean，主要是通过加载驱动程序、建立数据库连接的步骤得到 Connection 类的

一个对象。

（2）在 JSP 页面中通过 JSP 的 useBean 标记得到 DBbean 类中的连接，利用 request 对象的 getParameter()方法获得文本框中输入的数据，然后通过执行 SQL 语句获得查询结果，并将查询结果输出到表格中。具体代码如下：

```
<%
    try{
     Connection connection=con.getCon();                              //得到一个数据库连接
     Statement st=connection.createStatement();
     String sql ="select * from tb_booksell where selldate >= '"+request.getParameter("txt1")+"' and selldate<=
        '"+request.getParameter ("txt2")+"'order by selldate ";          //利用 SQL 语句进行查询
      ResultSet rs=st.executeQuery(sql);                               //执行 SQL 语句
     if(rs.next())
     {
    do{                                                                //循环输出查询的结果
%>
     <tr>
      <td width="10%" height="26" align="left" >  <%=rs.getString("id")%></td>
      <td width="37%" align="left" >   <%=rs.getString("bookname")%></td>
      <td width="18%" align="left" >       <%=rs.getString("author")%></td>
      <td width="16%" align="left" >        <%=rs.getString("price")%></td>
      <td width="19%" align="left" >    <%=rs.getString("selldate")%></td>
     </tr>
<%
    }while(rs.next());
    }
    else
    {
        out.println("<br><strong><center>没有你要检索的数据!</center><strong>");
    }
  } catch(Exception ex) {System.out.println(ex.getMessage());}%>
```

■ 秘笈心法

心法领悟 111：注意大于号、小于号的使用。

除了本例所讲的“>=”、“<=”之外，也可以利用“>”、“<”来实现查询指定时间段的数据。在使用小于（<）和大于（>）运算符时，由于这些运算符是非包含性的，所以会返回与本例不同的结果。

实例 112	计算两个日期间的月份数 光盘位置：光盘\MR\04\112	中级 实用指数：★★☆

■ 实例说明

本实例实现根据商品的上市时间统计商品的上市月数。运行程序，将会看到所有商品的上市时间以及到当前日期为止的上市月数，如图 4.26 所示。

商品上市时间统计

商品名称	商品价钱	上市时间	上市的月数
xx洗衣机	1280	2008-09-07	27 个月
xx风扇	329	2008-12-09	24 个月
xx空调	2000	2009-08-29	16 个月
xx电磁炉	876	2009-01-08	23 个月
xx笔记本	5600	2009-09-08	15 个月
xx豆浆机	765	2010-01-09	11 个月
xx电冰箱	2800	2010-09-08	3 个月

图 4.26 计算两个日期间的月份数

■

要实现对上市月数的计算，需要对获取的字符串调用 String 对象的 split()方法。

split()方法可以按指定的分隔字符或字符串将字符串内容进行分隔，其语法如下：

```
str.split(regex)
```

参数说明

❶ str：一个字符串对象。

❷ regex：分隔字符串的分隔符，也可以使用正则表达式分隔字符串。

str 字符串中没有统一的分隔符，可以使用“|”定义多个分隔符。例如“,|-|！”分别以“,”、“-”、“！”作为分隔符。

设计过程

（1）创建数据库连接类 DBbean，主要是通过加载驱动程序、建立数据库连接的步骤得到 Connection 类的一个对象。

（2）用 Date 类实例化一个 Date 对象，同时利用 SQL 语句获取商品的上市时间，通过相应的格式化日期方法对二者的数据进行格式化，然后利用 String 对象的 split()方法对其进行分隔，把分隔后的数据通过相应的计算作为一个字段输出到表格中。具体代码如下：

```
<%
      SimpleDateFormat format=new SimpleDateFormat("yyyy-MM-dd");          //对当前时间进行格式化
      String date2= format.format(new java.util.Date());
      try{
      Connection connection=con.getCon();                                  //得到一个数据库连接
      Statement st=connection.createStatement();
      String sql ="select * from tb_goods";                                //利用 SQL 语句进行查询
      ResultSet rs=st.executeQuery(sql);                                   //执行 SQL 语句
      if(rs.next())
      {
       do{
        String date1=rs.getString("marketdate");
        String[]   sdate1=date1.split("-");                                //用“-”分隔从数据库中取出的数据
        String[]   sdate2=date2.split("-");                                //用“-”分隔当前系统时间
        int count1=(Integer.parseInt(sdate2[0])-Integer.parseInt(sdate1[0]))*12
        +(Integer.parseInt(sdate2[1])-Integer.parseInt(sdate1[1]));       //对分隔出来的数据进行相应的计算
%>
      <tr>
        <td width="132" align="left"><div align="center"><%=rs.getString("goodsname")%></div></td>
        <td width="84" align="left"><div align="center"><%=rs.getString("goodsprice")%></div></td>
        <td width="118" align="left"><div align="center"><%=rs.getString("marketdate")%></div></td>
        <td width="154" align="left"><div align="center"><%=count1%>  个月</div></td>
      </tr>
      <%
    }while(rs.next());
    }
    else
    {
       out.println("<br><strong><center>没有你要检索的数据!</center><strong>");
    }
  } catch(Exception ex) {System.out.println(ex.getMessage());}%>
```

心法领悟 112：String 对象分隔符的使用。

String 对象调用 split()方法对指定数据进行分隔，得到的是一个 String 类型的数组，可以通过数组的下标取得分隔后的每个数据。

第 5 章

复杂查询技术

- 使用子查询
- 多表连接查询
- 嵌套查询
- 常见谓词的使用

5.1　使用子查询

实例 113　将子查询作为表达式

光盘位置：光盘\MR\05\113

高级

实用指数：★★★★

实例说明

将子查询作为表达式，是指在 SELECT 语句指定查询元素时，仍然使用 SELECT 查询语句。这种查询方式很常见，是程序员必须掌握的一种查询技巧。本实例将通过在子查询中嵌入表达式，实现查询学生总成绩与全校平均总成绩，并计算两者之差，如图 5.1 所示。

学　生　信　息　查　询

当前位置>>>学生成绩与全校平均分对比

选择查询条件：　所有学生信息　查询

ID	姓名	总成绩	全校平均总成绩	与全校平均分相差
1	张*	295.0	222.0	73.0
2	王*强	245.0	222.0	23.0
3	李*	259.0	222.0	37.0
19	学生A	251.0	222.0	29.0
5	李四	177.0	222.0	-45.0
20	学生B	162.0	222.0	-60.0
21	学生C	167.0	222.0	-55.0

图 5.1　学生成绩与全校平均分对比

关键技术

本实例的子查询是应用在 SELECT 子句中的。将子查询应用在 SELECT 子句中，其查询结构就可以以表达式的形式出现。在应用子查询时有一些控制规则，了解这些规则有助于更好地掌握子查询的应用。

（1）由比较运算符引入的内层查询 SELECT 列表或 IN 只包括一个表达式或列名。在外层语句的 WHERE 子句中命名的列必须能与查询 SELECT 列表中命名的列连接兼容。

（2）由不可更改的比较运算符引入的子查询（比较运算符后面不跟关键字 ANY 和 ALL）不能包括 GROUP BY 或 HAVING 子句，除非预先确定了组或单个的值。

（3）由 EXISTS 引入的 SELECT 列表一般都由星号（*）组成，而不必指定具体的列名，也可以在嵌套子查询 WHERE 子句中限定行。对于 EXISTS 引入的子查询，SELECT 列表规则和标准选择列表中的规则是一样的。

（4）子查询不能在内部处理它们的结果，也就是说，子查询不能包括 ORDER BY 子句。可选择的 DISTINCT 关键字可有效地对子查询结果进行排序，因为一些系统会通过首先将结果排序来消除重复记录。

（1）本实例实现了两大功能，分别为显示全部学生总成绩及学生总成绩与全校平均总成绩之差。首先创建类 JDBCconnection，在该类中定义获取数据库连接、关闭数据库连接、获取查询结果集的相关方法。具体代码参见配书光盘中的源程序。

（2）在项目的 index.jsp 页面中，定义表格显示学生成绩表中的全部信息。具体代码如下：

```
<td height="100" colspan="3" align="center"><table width="100%" height="80%" border="0">
 <tr>
     <td width="15%" height="25" align="center" bordercolor="#666666"> ID </td>           //定义表格内容
     <td width="15%" align="center" bordercolor="#666666"> 姓名 </td>
     <td width="27%" align="center" bordercolor="#666666"> 语文成绩 </td>
     <td width="26%" align="center" bordercolor="#666666"> 数学成绩 </td>
     <td width="27%" align="center" bordercolor="#666666"> 英语成绩 </td>
 </tr>
<%
String sql = "select StuID,StuName,Languages,Mathematics,English from tb_student";        //定义查询 SQL 语句
JDBCconnection db = new JDBCconnection();                                                 //创建保存有查询方法的 JDBCconnection 对象
ResultSet rs = db.selectAll(sql);                                                          //获取查询结果集
while (rs.next()) {                                                                        //循环遍历查询结果集
%>
 <tr align=center bgcolor="#f1f3f5">
```

```
    <td height="30" align="center" bordercolor="#666666"><%=rs.getString("StuID")%></td>
    <td bordercolor="#666666"><%=rs.getString("StuName")%></td>                    //将查询结果显示在页面中
    <td bordercolor="#666666"><%=rs.getString("Languages")%></td>
    <td bordercolor="#666666"><%=rs.getString("Mathematics")%></td>
    <td bordercolor="#666666"><%=rs.getString("English")%></td>
  </tr>
  <%}%>
```

（3）定义名为 StuServlet 的 Servlet，实现处理请求。当用户选择查询“学生成绩与全校平均分对比”时，提交至该 Servlet。在该 Servlet 中，首先定义 SQL 语句来查询学生成绩与全校平均分的对比，然后将查询结果返回。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String tiaojian = request.getParameter("tiaojian");                     //获取用户提交的请求信息
        if ("selectall".equals(tiaojian)) {                                     //判断请求内容
            request.getRequestDispatcher("index.jsp").forward(request,
                    response);                                                  //将请求转发至 index.jsp 页
        }
        if ("selects".equals(tiaojian)) {
            String sql = "select StuId,StuName,(Languages+Mathematics+English) " +
                    "Total,round((select avg(Languages+Mathematics+English)from tb_student),0) " +
                    "Averages,round(((Languages+Mathematics+English)-" +
                    "(select avg(Languages+Mathematics+English)from tb_student)),0) Average from tb_student";   //定义查询 SQL 语句
            JDBCconnection db = new JDBCconnection();                           //获取数据库连接
            ResultSet rs = db.selectAll(sql);                                   //调用执行查询方法
            request.setAttribute("rs", rs);                                     //将查询结果保存在 request 对象中
            request.getRequestDispatcher("stuSelect.jsp").forward(request,
                    response);
        }
}
```

秘笈心法

心法领悟 113：'a'与"a"的区别。

从表面上看，'a'与"a"所表示的值是相等的，但是这两者却有着本质上的区别。

其中，'a'表示基本数据类型 char 类型变量，而"a"表示字符串对象。对基本数据类型进行比较可以使用符号“==”，对字符串进行比较则需使用 equals()方法。

实例 114 用子查询作为派生表

光盘位置：光盘\MR\05\114

高级

实用指数：★★★

实例说明

在实际应用中，经常使用子查询作为派生表，就是将查询结果集作为一个表使用。本实例将实现在战士训练表中查询第 3 次射击成绩大于 8 环的战士的信息，运行结果如图 5.2 所示。

战 士 训 练 信 息 查 询

当前位置>>>第三次射击成绩大于8环的战士信息

选择查询条件：所有战士训练信息 查询

编号	姓名	第1次射击成绩	第2次射击成绩	第3次射击成绩
1	张*	9.7	9.9	10.0
3	王五	8.5	8.6	8.7

图 5.2　查询第 3 次射击成绩大于 8 环的战士信息

子查询是一个用于处理多表操作的附加方法。

语法如下：

```
(SELECT [ALL | DISTINCT]<select item list>
FROM <table list>
[WHERE<search condition>]
[GROUP BY <group item list>
[HAVING <group by search conditoon>]])
```

把子查询用作派生的表可以应用在很多方面，例如下面几个示例。

将分组统计数据作为派生表（将销售单按商品名称统计分组后查询销售数量大于 14 的商品）：

```
SELECT * FROM (SELECT proname, COUNT(*) AS sl FROM xsd GROUP By proname) WHERE (sl > 14)
```

将过滤数据作为派生表（对商品销售表中销售数量前 100 名进行分组统计）：

```
SELECT sl,COUNT(*) FROM (SELECT TOP 100 FROM T_ZDxxb ORDER BY zdbh) GROUP BYsl
```

将过滤数据作为派生表（统计客户表中未结账客户的欠款金额）：

```
SELECT name,SUM(xsje) FROM (SELECT * FROM kh WHERE NOT jz) GROUP BY name
```

设计过程

（1）本实例实现了两大功能，分别为查询所有战士训练信息和查询第 3 次射击成绩大于 8 环的战士信息。关于页面设计，参见配书光盘中的源程序。

（2）定义名为 SoldServlet 的 Servlet，在该 Servlet 中处理请求，并根据用户请求内容定义转发地址。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    String tiaojian = request.getParameter("tiaojian");                    //获取用户提交的请求信息
    if ("selectall".equals(tiaojian)) {                                     //判断用户提交的内容
        request.getRequestDispatcher("index.jsp")                           //如果用户选择查询全部信息，将请求转发至系统首页
                .forward(request, response);
    }
    if ("selects".equals(tiaojian)) {                                       //如果用户选择第 3 次射击大于 8 环的战士的信息
        String sql = "select T.SoldId,T.SoldName,T.FirstGun,T.SecondGun," +
                "T.ArtideGun from (select * from tb_soldiers where ArtideGun>8) as T";      //定义查询 SQL 语句
        JDBCconnection db = new JDBCconnection();                                           //定义保存查询方法的 JavaBean 对象
        ResultSet rs = db.selectAll(sql);                                                   //调用查询方法
        request.setAttribute("rs", rs);                                                     //将查询结果集保存在 request 对象中
        request.getRequestDispatcher("soldSelect.jsp").forward(request,
                response);                                                                  //定义请求地址
    }
}
```

秘笈心法

心法领悟 114：必须为派生表起别名。

在本实例中，是将一个查询结果作为另一个查询所操作的表。在查询时需要注意，由于 SELECT 子句一定要给出一个表，所以作为 SELECT 语句派生的表一定要给出一个别名。

实例 115	通过子查询关联数据 光盘位置：光盘\MR\05\115	高级 实用指数：★★★

实例说明

本实例利用 EXISTS 谓词引入子查询，将学生表和学生成绩表关联起来，查询英语成绩大于 90 分的学生姓名、学科、家庭住址等信息，运行结果如图 5.3 所示。

学 生 信 息 查 询

当前位置>>>查询英语成绩大于90的学生信息

选择查询条件：所有学生信息　查询

姓名	学科	家庭住址
格雷	计算机学院	吉林省长春市
刘静	理工大学	江苏省海门市
封号	机械学院	北京市

图 5.3　查询英语成绩大于 90 的学生信息

在某些情况下，只要子查询返回一个真值或假值，只考虑是否满足谓词条件，数据内容本身并不重要。此时可用 EXISTS 谓词来定义子查询。如果子查询返回一行或多行，EXISTS 谓词为真，否则为假。要使 EXISTS 谓词起作用，应该在子查询中建立查询条件以匹配子查询连接起来的两个表中的值。

语法如下：

```
EXISTS subquery
```

参数说明

subquery：一个受限的 SQL 语句（不允许有 COMPUTE 子句和 INTO 关键字）。

其结果类型为 Boolean，如果子查询包含行，则返回 true。

注意：EXISTS 谓词子查询中的 SELECT 子句中可使用任何列名，也可以使用任何多个列。这种谓词只注重是否返回行，而不注重行的内容，用户可以指定列名或者只使用一个“*”。

设计过程

（1）在项目中定义类 JDBCconnection，在该类中定义获取数据库连接方法（具体代码参见配书光盘中的源程序）；然后定义查询方法，实现获取英语成绩大于 90 分的学生信息。具体代码如下：

```
public ResultSet getMessage() {
    ResultSet rest = null;
    con = creatConnection();                                    //获取与数据库的连接
    try {
        Statement statement = con.createStatement();            //获取 Statement 对象
        String sql = "select name,college,address from tb_Stu I where exists " +
                "(select name from tb_grades M where M.name=I.name and english >90)";    //定义查询语句
        rest = statement.executeQuery(sql);                     //执行查询语句获取查询结果集
    } catch (Exception e) {
        e.printStackTrace();
    }
    return rest;                                                //返回查询结果
}
```

（2）定义名为 StuServlet 的 Servlet，在该类中获取用户提交的请求，并根据请求作出相应的处理。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    String tiaojian = request.getParameter("tiaojian");         //获取用户提交的请求内容
    if ("selectall".equals(tiaojian)) {                         //判断请求提交内容
        request.getRequestDispatcher("index.jsp").forward(request,
                response);                                      //给出请求转发地址
    }
    if ("selects".equals(tiaojian)) {
        JDBCconnection db = new JDBCconnection();               //创建包含有查询方法的类对象
        ResultSet rs = db.getMessage();                         //调用查询英语成绩大于 90 分的学生信息
        request.setAttribute("rs", rs);                         //将查询方法保存在 request 对象中
        request.getRequestDispatcher("stuSelect.jsp").forward(request,
                response);                                      //设置请求转发地址
    }
}
```

秘笈心法

心法领悟 115：为什么要进行丢失精度的类型转换。

由高精度类型向低精度进行转换时，会出现由于精度丢失而使数据不完整的情况。从表面上来看，这种类型转换没有什么优势，但是 Java 为什么还要设置这种机制呢？这其实是为了满足特殊的用户编程需求。例如，现有一个程序的数据为 12.56，而后来程序需要的数据是不包含小数的，此时使用其他的算法进行处理都是很复杂的，而使用强制数据类型转换就可以很轻松地实现想要的结果。

实例 116 使用 IN 谓词限定查询范围

光盘位置：光盘\MR\05\116

中级

实用指数：★★★★☆

实例说明

IN 谓词允许测试实际值或表达式的值。当将 IN 谓词嵌入子查询时，就是告诉数据库管理系统执行一种“子

查询成员测试”。IN 语句可以替代 WHERE 子句中的表达式用以限定查询的范围。本实例将实现在子查询中嵌入 IN 谓词，查询英语成绩为优的学生信息，如图 5.4 所示。

学　生　信　息　查　询

当前位置>>>英语成绩为优的学生信息

选择查询条件：所有学生信息　查询

ID	姓名	语文成绩	数学成绩	英语成绩
1	张*	98.0	97.0	100.0
3	李*	75.0	85.0	99.0
19	学生A	56.0	100.0	95.0

图 5.4　使用 IN 谓词限定查询范围

关键技术

IN 谓词用于确定给定的值是否与子查询或列表中的值相匹配，常用于引入子查询。

语法：

```
test_expression [ NOT ] IN
    (
        subquery
        | expression [ ,...n ]
    )
```

参数说明

❶ test_expression：任何有效的 SQL 表达式。

❷ subquery：包含某列结果集的子查询，该列必须与 test_expression 具有相同的数据类型。

❸ expression [,...n]：一个表达式列表，用来测试是否匹配。所有的表达式必须和 test_expression 具有相同的数据类型。

如果 test_expression 与 subquery 返回的任何值相等，或与逗号分隔的列表中的任何 expression 相等，那么结果值就为 true，否则为 false。

设计过程

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库类。具体代码参见配书光盘中的源程序。

（2）在项目中定义名为 StuServlet 的 Servlet，在该 Servlet 中处理用户提交请求，并根据用户提交内容进行相应处理。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        String tiaojian=request.getParameter("tiaojian");                //获取用户提交请求信息
        if("selects".equals(tiaojian)){                                  //判断请求信息内容
                String sql="select StuID,StuName,Languages,Mathematics," +
                        "English from tb_student where StuId in " +
                        "(select StuId from tb_student where English>85)";   //定义查询 SQL 语句
                JDBCconnection db=new JDBCconnection();                  //创建保存有操作数据库的类对象
                ResultSet rs =db.selectAll(sql);                         //调用执行查询方法
                request.setAttribute("rs", rs);                          //将查询方法保存在 request 对象中
                request.getRequestDispatcher("stuSelect.jsp").forward(request, response);  //设置转发地址
        }
        if ("selectall".equals(tiaojian)) {
                request.getRequestDispatcher("index.jsp").forward(request,
                        response);
        }
}
```

心法领悟 116：if 语句与 switch 语句的使用区别。

if 条件语句与 switch 多分支语句的功能基本类似，都可以实现当程序满足一定的条件后，执行相关的代码。

对于 if 语句，如果要判断多重条件，就需要使用 if else 语句。这样看来，当条件过多时，使用 switch 语句要简便一些；如果选择条件少于 3 条，使用 if 条件语句则更简便一些。因此，应根据程序的不同要求来选择适当的语句。

实例 117 使用 NOT IN 子查询实现差集运算

光盘位置：光盘\MR\05\117

高级

实用指数：★★★★

实例说明

上面的实例中为大家介绍了使用 IN 关键字进行查询，下面将讲解如何使用 NOT IN 子查询实现两张表中数据的差集运算。本实例将实现查询学生表与学生成绩表，并使用 NOT IN 关键字查询在学生成绩表没有成绩的学生的信息，运行结果如图 5.5 所示。

没 有 成 绩 的 学 生

当前位置>>>没有成绩的学生信息查询

选择查询条件： 所有成绩查询 查询

编号	姓名	学院	专业	地址
5	冯洁	机械学院	汽车制造	北京市
6	孙晓	外国语	英语	吉林省长春市

图 5.5 没有成绩的学生信息查询

关键技术

带 NOT IN 谓词的查询语法如下：

```
WHERE 查询表达式 NOT IN 子查询
```

NOT IN 和 IN 查询过程相似，可参考实例 116。

注意：当子查询存在 null 值时，应避免使用 NOT IN，因为当子查询的结果包括了 null 值的列表时，将把 null 值当成一个未知数据，而不会存在查询值不在列表中的记录。

设计过程

在项目中创建类 JDBCconnection，在该类中定义操作数据库的方法（有关获取数据库连接等方法，不是本实例重点，这里不再赘述，读者可参考配本光盘中的源程序）；然后查询在学生成绩表中没有包含的学生信息，并将查询结果以 List 形式返回。具体代码如下：

```
public List getNotIn() {
    List list = new ArrayList();                                          //定义用于保存返回值的 List 集合
    con = creatConnection();                                              //获取数据库连接
    try {
        Statement staement = con.createStatement();
        String sql = "select * from tb_Stu where name not in (select name from tb_grades)";   //定义查询数据的 SQL 语句
        ResultSet set = staement.executeQuery(sql);                       //执行查询语句，返回查询结果集
        while (set.next()) {                                              //循环遍历查询结果集
          Stu stu = new Stu();                                            //创建与数据表对应的 JavaBean 对象
          stu.setId(set.getInt(1));                                       //应用查询结果设置对象属性
          stu.setName(set.getString(2));
          stu.setColleg(set.getString(3));
          stu.setSpeciality(set.getString(4));
          stu.setAddress(set.getString(5));
          list.add(stu);                                                  //将 JavaBean 对象添加到 List 集合中
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return list;                                                          //返回 List 集合
}
```

秘笈心法

心法领悟 117：Transact-SQL 中的数据类型转换。

与 Java 语言一样，在 Transact-SQL 中也支持数据类型转换。Transact-SQL 中的数据类型转换有两种，分别

介绍如下。

- ❑ 隐性转换：对用户是不可见的。SQL Server 自动将数据从一种数据类型转换成另一种数据类型。例如，如果一个 smallint 变量和一个 int 变量相比较，这个 smallint 变量在比较前即被隐性转换成 int 变量。
- ❑ 显式转换：使用 CAST 或 CONVERT 函数。CAST 或 CONVERT 函数可将数值从一种数据类型（局部变量、列或其他表达式）转换为另一种数据类型。

实例 118　使用 NOT IN 子查询实现反向查询　中级

光盘位置：光盘\MR\05\118　　实用指数：★★★

实例说明

前文提到，谓词 IN 可以查询在某范围内的数据。该谓词与谓词 NOT 一起使用，可以查询不在指定范围内的数据。本实例将使用 NOT IN 组合查询，实现查询平均销量低于所有平均销量的图书信息，运行结果如图 5.6 所示。

图 书 销 量 信 息 查 询

当前位置>>>平均销量低于所有平均销量的图书信息

选择查询条件：平均销量低于所有平均销量的图书信息　查询

编号	图书名称	一月份销量	二月份销量	三月份销量
2	C#从入门到精通	800	1000	900
4	ASP从入门到精通	500	600	700

图 5.6　查询平均销量低于所有平均销量的图书信息

关键技术

NOT 与 IN 两个关键字联用表示不在列表范围内的数据。查询表达式可以是常量或列名，而列表可以是一组常量，更多情况下子查询将列表值放在圆括号内。

注意： null 值进行取反，结果仍是 null，这一点在具体应用中经常会被忽略。

（1）本实例实现了查询所有图书销量信息与平均销量低于所有平均销量的图书信息。首先创建保存操作数据库类 JDBCconnection，该类中的方法参见配书光盘中的源程序，这里不再赘述。

（2）在项目的 index.jsp 页面中，定义表格显示所有图书销量信息。代码如下：

```
<td height="100" colspan="3" align="center"><table width="98%" height="80%" border="0">
    <tr>
        <td width="15%" height="25" align="center" bordercolor="#666666">编号</td>
        <td width="25%" align="center" bordercolor="#666666">图书名称</td>
        <td width="20%" align="center" bordercolor="#666666">一月份销量 </td>
        <td width="20%" align="center" bordercolor="#666666">二月份销量 </td>
        <td width="20%" align="center" bordercolor="#666666">三月份销量 </td>
    </tr>
    <%
            String sql = "select BookId,BookName,January,February,March from tb_book";     //定义查询所有图书信息 SQL 语句
            JDBCconnection db = new JDBCconnection();                                       //定义操作数据库的类对象
            ResultSet rs = db.selectAll(sql);                                               //调用获取查询结果集方法
            while (rs.next()) {                                                             //循环遍历查询结果集
        %>
    <tr align=center bgcolor="#f1f3f5">
        <td height="30" align="center" bordercolor="#666666"><%=rs.getString("BookId")%></td>   //在页面中显示查询结果
        <td bordercolor="#666666"><%=rs.getString("BookName")%></td>
        <td bordercolor="#666666"><%=rs.getString("January")%></td>
        <td bordercolor="#666666"><%=rs.getString("February")%></td>
        <td bordercolor="#666666"><%=rs.getString("March")%></td>
    </tr>
    <%}%>
```

（3）在页面中定义名为 bookServlet 的 Servlet，获取用户提交的请求，并对请求作出相应的处理。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
```

```
        String tiaojian = request.getParameter("tiaojian");                //获取用户提交内容
        if ("selects".equals(tiaojian)) {                                  //判断用户提交内容
            String sql = "select BookId,BookName,January,February,March" +
                    " from tb_book where BookId not in" +
                    "(select BookId from tb_book where (January+February+March)" +
                    ">(select avg(January+February+March)from tb_book))";    //定义查询 SQL 语句
            JDBCconnection db = new JDBCconnection();                      //定义操作数据库的类对象
            ResultSet rs = db.selectAll(sql);                              //调用操作数据库的方法
            request.setAttribute("rs", rs);                                //将查询结果保存在 request 对象中
            request.getRequestDispatcher("bookSelect.jsp").forward(request,
                    response);                                             //设置请求转发地址
        }
        if ("selectall".equals(tiaojian)) {                                //判断用户提交请求
            request.getRequestDispatcher("index.jsp")
                    .forward(request, response);
        }
}
```

秘笈心法

心法领悟 118：数组最大容量问题。

数组的最大容量是声明数组时必须要考虑的，因为数组一旦被声明，其最大容量就固定了，是不可改变的。因此，为了防止出现容量不足的现象，在数组声明之初一定要考虑数组的最大容量问题，以避免程序在后续运行中出现不必要的麻烦。

实例 119 实现笛卡儿乘积查询

光盘位置：光盘\MR\05\119

高级

实用指数：★★★★☆

实例说明

在多表连接查询中，有一种非常重要的查询方式——笛卡儿乘积查询，即将两张表交叉连接实现查询。本实例通过将职位表与员工表进行笛卡儿乘积连接来实现员工信息查询，运行结果如图 5.7 所示。

员 工 信 息 查 询

当前位置>>>笛卡尔乘积查询

选择查询条件： 所有员工及职位信息 查询

ID	姓名	部门	职位	工资
1	张*	销售部	经理	3500.0
2	李*	客服部	经理	2500.0
3	王*	销售部	经理	2500.0
4	赵*	客服部	经理	1500.0
5	孙**	技术部	经理	2700.0
1	张*	销售部	主任	3500.0
2	李*	客服部	主任	2500.0
3	王*	销售部	主任	2500.0
4	赵*	客服部	主任	1500.0
5	孙**	技术部	主任	2700.0
1	张*	销售部	员工	3500.0
2	李*	客服部	员工	2500.0
3	王*	销售部	员工	2500.0
4	赵*	客服部	员工	1500.0
5	孙**	技术部	员工	2700.0

图 5.7 笛卡儿乘积查询

关键技术

笛卡儿乘积查询实现了两张表之间的交叉连接，在查询语句中没有 WHERE 查询条件，返回到结果集中的数据行数等于一个表中符合查询条件的数据行数乘以第 2 个表中符合条件的数据行数。

笛卡儿乘积的关键字是 CROSS JOIN。例如，用户信息表中有 2 条数据，职工信息表中有 4 条数据，当这两张表应用笛卡儿乘积进行查询时，查询的结果就是 8 条（2×4）。

（1）创建操作数据库类 JDBCconnection，在该类中定义操作数据库的各种方法，具体代码参见配书光盘中的源程序。

（2）创建名为 WorkServlet 的 Servlet，在该 Servlet 中获取用户提交的请求，并根据请求进行相应的处理。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
```

```
        String tiaojian = request.getParameter("tiaojian");                    //获取用户提交的请求
        if ("selects".equals(tiaojian)) {                                       //判断用户提交内容
            JDBCconnection db = new JDBCconnection();                           //创建保存有操作数据库的类对象
            String sql = "select EmpId,EmpName,Depatment,JobTitle,Wages from tb_employees a cross join tb_position b";//定义查询 SQL 语句
            ResultSet res = db.selectAll(sql);                                  //调用查询方法
            request.setAttribute("rs", res);                                    //将查询结果保存在 request 对象中
            request.getRequestDispatcher("jobSelect.jsp").forward(request,
                    response);                                                  //定义请求转发地址
        }
        if ("selectall".equals(tiaojian)) {                                     //判断用户提交内容
            request.getRequestDispatcher("index.jsp").forward(request,
                    response);
        }
}
```

秘笈心法

心法领悟 119：确定表的列。

在进行多表查询时需要注意，由于多个表可能会出现相同的字段，因此在指定查询字段时，最好为重复的字段起别名，以方便区分。

实例 120　比较运算符引入子查询

光盘位置：光盘\MR\05\120

高级

实用指数：★★★★

实例说明

比较运算符在查询中应用得较为广泛。如果将比较运算符两端的数据定义好，程序就不会太灵活；而如果通过子查询来指定查询条件，则会增加程序的灵活性。本实例通过将比较运算符引入子查询来实现查询第一次射击成绩大于平均第一次射击成绩的战士信息，运行结果如图 5.8 所示。

战　士　训　练　信　息　查　询

当前位置>>>所有战士训练信息

选择查询条件：第一次射击成绩大于平均第一次射击成绩的战士信息　查询

编号	姓名	第一次射击成绩	第二次射击成绩	第三次射击成绩
1	张*	9.7	9.9	10.0
2	李四	5.4	8.5	6.7
3	王五	8.5	8.6	8.7
4	赵六	9.5	6.0	5.4

图 5.8　查询第一次射击成绩大于平均第一次射击成绩的战士信息

关键技术

在比较运算符中使用子查询，就是将比较运算符“<”或“>”后的表达式以一个子查询来代替。在使用子查询时要注意，子查询要使用“()”括起来，否则会出现 SQL 异常提示。

（1）在项目中创建类 JDBCconnection，在该类中定义获取数据库连接、关闭数据库连接、获取查询结果集等方法。

（2）在项目中创建名为 SoldServlet 的 Servlet，在该 Servlet 中处理用户提交的请求。本实例实现了两大查询功能，分别为查询所有的战士信息与查询第一次射击成绩大于平均第一次射击成绩的战士信息。代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String tiaojian=request.getParameter("tiaojian");                       //获取用户提交的请求信息
        if("selects".equals(tiaojian)){                                         //判断用户提交的请求内容
            String sql="select SoldId,SoldName,FirstGun,SecondGun,ArtideGun" +
                    " from tb_soldiers where FirstGun >" +
                    " (select avg(FirstGun) from tb_soldiers)";                 //定义查询 SQL 语句
            JDBCconnection db=new JDBCconnection();                             //定义操作数据库的类对象
            ResultSet rs=db.selectAll(sql);                                     //调用获取查询结果集方法
            request.setAttribute("rs", rs);                                     //将查询结果保存在 request 对象中
            request.getRequestDispatcher("soldSelect.jsp").forward(request, response);    //设置请求转发地址
```

```
        }
        if("selectall".equals(tiaojian)){
            request.getRequestDispatcher("index.jsp").forward(request, response);        //设置请求转发地址
        }
}
```

■ 秘笈心法

心法领悟 120：for 循环语句与 while 循环语句的使用区别。

本实例在遍历查询结果集时使用的是 while 循环。除此之外，还有一种十分常见的循环语句，就是 for 循环，for 语句与 while 语句都是很重要的循环语句，执行的效果是相同的，都是在满足一定的条件下使程序重复地执行，那么对一个普通循环来说，是使用 for 循环还是 while 循环呢？来看一下这两种循环语句的区别就会得到答案。for 循环语句主要针对有限循环而言，也就是说当循环有上限时，一般使用 for 循环；while 循环语句则针对无限循环的代码而言，也就是当程序没有明确的上限时，一般使用 while 循环。

实例 121 在查询中使用聚合函数

光盘位置：光盘\MR\05\121

高级

实用指数：★★★★

■ 实例说明

在实际开发中，子查询有非常重要的作用，灵活地运用子查询可解决很多问题。本实例将在子查询中使用聚合函数，实现查询总成绩大于全连平均总成绩的战士信息，运行结果如图 5.9 所示。

战 士 训 练 信 息 查 询

当前位置>>>总成绩大于全连平均总成绩的战士信息

选择查询条件： 总成绩大于全连平均总成绩的战士信息 查询

编号	姓名	第一环成绩	第二环成绩	第三环成绩
1	张*	9.7	9.9	10.0
3	王五	8.5	8.6	8.7

图 5.9　查询总成绩大于全连平均总成绩的战士信息

■ 关键技术

本实例在查询战士的平均成绩时，使用的是 AVG 函数，该函数具体语法参见实例 088，这里不再赘述。

■

（1）在项目中创建类 JDBCconnection，在该类中定义获取数据库连接、关闭数据库连接、获取查询结果集等方法，具体代码参见配书光盘中的源程序。

（2）本实例实现了两大功能，分别为查询所有战士的训练成绩与总成绩大于全连平均总成绩的战士信息。在项目的 index.jsp 页面中，定义表格显示所有战士的训练成绩。具体代码如下：

```
<td height="100" colspan="3" align="center"><table width="98%" height="80%" border="0">
  <tr>
        <td width="15%" height="25" align="center">编号</td>
        <td width="15%" align="center">姓名</td>
        <td width="27%" align="center">第一环成绩</td>
        <td width="26%" align="center">第二环成绩</td>
        <td width="27%" align="center">第三环成绩</td>
 </tr>
 <%
        String sql = "select SoldId,SoldName,FirstGun,SecondGun,ArtideGun from tb_soldiers";      //定义查询数据的 SQL 语句
        JDBCconnection db = new JDBCconnection();                                                  //定义保存有操作数据库的类对象
        ResultSet rs = db.selectAll(sql);                                                          //调用获取数据库连接方法
              while (rs.next()) {                                                                  //循环遍历查询结果集
 %>
<tr align=center bgcolor="#f1f3f5">
   <td height="30" align="center" bordercolor="#666666"><%=rs.getString("SoldId")%></td>
   <td bordercolor="#666666"><%=rs.getString("SoldName")%></td>                                   //将查询结果显示在页面中
   <td bordercolor="#666666"><%=rs.getString("FirstGun")%></td>
   <td bordercolor="#666666"><%=rs.getString("SecondGun")%></td>
```

```
  <td bordercolor="#666666"><%=rs.getString("ArtideGun")%></td>
</tr>
<%}%>
```

（3）创建名为 SoldServlet 的 Servlet，在该 Servlet 中获取用户请求，并作出相应的处理。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
      String tiaojian=request.getParameter("tiaojian");                          //获取用户提交的内容
      if("selects".equals(tiaojian)){                                            //判断用户提交的内容
            String sql="select SoldId,SoldName,FirstGun," +
                        "SecondGun,ArtideGun from tb_soldiers where " +
                        "(FirstGun+SecondGun+ArtideGun)>(select " +
                        "avg(FirstGun+SecondGun+ArtideGun) from tb_soldiers)";   //定义查询 SQL 语句
            JDBCconnection db=new JDBCconnection();                              //定义保存有操作数据库的类对象
            ResultSet rs=db.selectAll(sql);                                      //调用查询数据方法
            request.setAttribute("rs", rs);                                      //将数据保存在 request 对象中
            request.getRequestDispatcher("soldSelect.jsp").forward(request, response);   //定义请求转发地址
      }
      if("selectall".equals(tiaojian)){
            request.getRequestDispatcher("index.jsp").forward(request, response);
      }
}
```

秘笈心法

心法领悟 121：获取数据库信息。

在 JDBC 中提供了 ResultSetMetaData 接口，该接口用来获取数据表中每列的信息。可以通过 ResultSet 对象的 getMetaData()方法创建 ResultSetMetaData 对象。

```
ResultSet rs = rs = st.executeQuery("SELECT * FROM tableName");
ResultSetMetaData rsmd = rs.getMetaData();
```

下面介绍 ResultSetMetaData 接口中几个比较常用的方法。

- getColumnCount()方法：用于获取列数。实例代码如下：

```
ResultSetMetaData rsmd = rs.getMetaData();        //获取 ResultSetMetaData 对象
int count = rsmd.getColumnCount();                //获取表中列的数量
```

- getColumnName()方法：用于获取指定列的名称。该方法有一个 int 类型的参数。实例代码如下：

```
ResultSetMetaData rsmd = rs.getMetaData();        //获取 ResultSetMetaData 对象
String columnName = rsmd.getColumnName(1);        //获取第一列的名称
```

- getPrecision()方法：用于获取指定列的宽度。该方法有一个 int 型参数，用于指定列数。实例代码如下：

```
ResultSetMetaData rsmd = rs.getMetaData();        //获取 ResultSetMetaData 对象
int columnLength = rsmd.getPrecision(1);          //获取第一列宽度
```

实例 122　在删除数据时使用子查询

光盘位置：光盘\MR\05\122

高级　实用指数：★★★

实例说明

使用 DELETE 语句可以实现从数据库中删除记录。本实例将在删除语句中实现子查询，将员工表中不存在的部门信息删除，如图 5.10 所示。运行本实例，首先会将部门信息表中的全部内容显示在页面中；当用户单击“执行”按钮时，会将指定的内容删除。

部　门　信　息　查　询

当前位置>>>所有部门信息

执行条件：　删除在员工表中不存在的部门信息　执行

ID	部门名称	负责人
1	系统分析部	高岩
2	软件测试部	刘芳
3	设计部	蓝鹏
4	开发部	李玉

图 5.10　在删除数据时使用子查询

DELETE 语句可以和子查询联合使用，由子查询指定删除数据记录的条件。语法如下：

```
DELETE FROM tableName
WHERE search_condition IN|NOT IN(子查询)
```

参数说明

❶ tableName：指定要操作的数据表。

❷ search_condition：指定条件的数据列。

设计过程

在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，然后定义删除数据方法。具体代码如下：

```
public void deleteEmp() {
    con = creatConnection();                                                        //获取与数据库的连接
    try {
        Statement statement = con.createStatement();                                //获取 Statement 对象
        String sql = "delete from tb_mrdept where person not in (select name from tb_mremp )";  //定义删除数据的 SQL 代码
        statement.executeUpdate(sql);                                               //执行删除 SQL 语句
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 122：获取系统有效盘符。

不同计算机的系统盘符是不相同的，有的计算机有“C:”、“D:”、“E:”、“F:”盘，而有的计算机只有“C:”、“D:”盘。那么如何获取本地有效盘符呢？通过 File 类的 listRoots()方法即可获取本地有效磁盘。关键代码如下：

```
public File[] getRoot(){
    File[] roots = File.listRoots();
    return roots;
}
```

5.2 多表连接查询

实例 123 使用 UNION 运算符使学生档案归档

光盘位置：光盘\MR\05\123

中级

实用指数：★★★

实例说明

UNION 指的是并运算，即从两个或多个类似的结果集中选择行，并将其组合在一起形成一个单独的结果集。本实例利用 UNION 运算符将初中生表与高中生表进行归档，运行结果如图 5.11 所示。

学 生 档 案 信 息 查 询

当前位置>>>初高中学生档案整合信息

整合信息			
学生姓名	毕业院校	就读时所在地址	档案编号
张三	**市第一初中	长春绿园区	1001
李四	**市第二初中	长春宽城区	1002
张三	**高中	长春市二道区	1001
李四	**高中	长春市二道区	1002

学生档案信息： 返回

图 5.11 归档初中生表与高中生表

UNION 运算符主要用于将两个或更多查询的结果组合为单个结果集，该结果集包含联合查询中所有查询的全部行。在使用 UNION 运算符时应遵循以下准则：

- ❑ 在使用 UNION 运算符组合的语句中，所有选择列表的表达式数目必须相同（列名、算术表达式、聚集函数等）。

- 在使用 UNION 运算符组合的结果集中的相应列或个别查询中使用的任意列的子集必须具有相同的数据类型，并且两种数据类型之间必须存在可能的隐性转换或提供了显式转换。例如，在 datetime 数据类型的列和 binary 数据类型的列之间不可能存在 UNION 运算符，除非提供了显式转换；而在 money 数据类型的列和 int 数据类型的列之间可以存在 UNION 运算符，因为它们可以进行隐性转换。
- 利用 UNION 运算符组合的各语句中对应的结果集列出现的顺序必须相同，因为 UNION 运算符是按照各个查询给定的顺序逐个比较各列。
- UNION 运算符组合不同的数据类型时，这些数据类型将使用数据类型优先级的规则进行转换。例如，int 值转换成 float 值，因为 float 类型的优先权比 int 类型高。
- 通过 UNION 运算符生成的表中的列名来自 UNION 语句中的第 1 个单独的查询。若要用新名称引用结果集中的某列（例如，在 ORDER BY 子句中），必须按第一个 SELECT 语句中的方式引用该列。

设计过程

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的方法。具体代码参见配书光盘中的源程序。

（2）在项目中创建名为 StuServlet 的 Servlet，在该 Servlet 中获取用户提交的请求，并作出相应的处理。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String action=request.getParameter("action");                              //获取用户提交的请求
        if("select".equals(action)){                                                //判断用户的请求内容
            JDBCconnection db=new JDBCconnection();                                 //创建保存有操作数据库的类对象
            String sql = "select filenumber,name,junior,address from " +
                    "tb_junior union select filenumber,name,senior,address from tb_senior";   //定义查询 SQL 语句
            ResultSet rs=db.selectAll(sql);                                         //调用获取查询结果集方法
            request.setAttribute("rs", rs);                                         //将查询结果保存在 request 对象中
            request.getRequestDispatcher("stuSelect.jsp").forward(request, response);   //设置请求转发地址
        }
        if("selectall".equals(action)){
            request.getRequestDispatcher("index.jsp").forward(request, response);
        }
}
```

秘笈心法

心法领悟 123：使用 DOS 命令实现 MySQL 数据库还原。

对于 MySQL 数据库来说，实现数据还原很简单，在 DOS 命令行中直接使用 MySQL 命令即可。可以使用如下命令行：

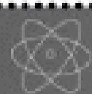
```
mysql –u root –p111 –h localhost db_wsy< c:\db_wsy.txt
```

参数说明

❶ -u：连接 MySQL 数据库的用户名。

❷ -p：连接 MySQL 数据库的密码。

❸ -h：连接 MySQL 数据库的主机名。



实例 124　内连接查询指定课程的教师信息

光盘位置：光盘\MR\05\124

中级　实用指数：★★★

实例说明

严格来讲，数据表是不允许出现冗余字段或经过计算可得到的数据列的。这样，如果程序员要获取某信息，就要通过查询语句来获取。因此，灵活地使用查询语句，对于一个程序员来说非常重要。本实例实现了使用内

连接（将教师表和课程表进行连接）查询指定课程的教师信息，运行结果如图 5.12 所示。

查询指定课程的教师信息

指定课程：所有课程教师 查询

姓名	性别	年龄	地址	科目
王*	男	45	长春市绿园区	数学
孙*	女	28	长春市绿园区	英语
张*	男	56	长春市绿园区	语文

图 5.12　使用内连接查询指定课程的教师信息

关键技术

内连接可分为等值连接、自然连接和不等连接。本实例采用的是等值连接。等值连接使用等号运算符比较被连接列的值，在查询结果中将列出连接表中的所有列，包括重复的列。等值连接返回所有连接表中具有匹配值的行。

等值连接查询的语法如下：

```
select fildList from table1 inner join table2 on table1.column = table2.column
```

参数说明

fildList：要查询的字段列表。

设计过程

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的方法，其中包含获取数据库连接、关闭数据库连接、获取查询结果集等方法。具体代码参见配书光盘中的源程序。

（2）在页面中将查询结果以表格的形式显示，具体代码如下：

```
<%
        String tiaojian=null;
        if(request.getParameter("tiaojian")!=null)                                  //判断用户是否单击了“提交”按钮
                tiaojian=request.getParameter("tiaojian");                          //获取用户提交的内容
        if(tiaojian==null||tiaojian.equals("")){
                tiaojian="1,2,3";
        }
        String sql = "select t.name,t.sex,t.age,t.address"+
        ",c.cname from tb_teacher t inner join tb_course c on t.cid=c.id and c.id in("+tiaojian+")";   //定义查询 SQL 语句
        JDBCconnection db = new JDBCconnection();                                  //创建包含有查询数据的类对象
        ResultSet rs = db.selectAll(sql);                                          //调用获取查询结果集方法
        while(rs.next()){                                                          //循环遍历查询结果集
%>
<tr align=center bgcolor="#f1f3f5">
        <td height="30" align="center"><%=rs.getString(1) %></td>                  //将查询结果显示在页面中
        <td><%=rs.getString(2) %></td>
        <td><%=rs.getString(3) %></td>
        <td><%=rs.getString(4) %></td>
        <td><%=rs.getString(5) %></td>
</tr>
<%}%>
```

秘笈心法

心法领悟 124：使用 Java 定时器。

定时器在实际开发中应用较为广泛。实现 Java 定时器可以使用 java.util.Timer 类来完成，其实例可以调用 schedule()方法。例如：

```
Timer t = new Timer();
t.schedule(myTimeTask,date,timestamp);
```

实例 125　左外连接查询员工信息

光盘位置：光盘\MR\05\125

高级

实用指数：★★★★

实例说明

实例 124 介绍了使用内连接查询数据，下面讲解另一种查询方式，即外连接。外连接与内连接的区别在于，

内连接只返回两张表相匹配的数据；而外连接是对内连接的扩展，可以使查询更具完整性，不会丢失数据。本实例实现了部门表左外连接员工表，运行结果如图 5.13 所示。

左外连接查询员工信息

编号	姓名	性别	年龄	职位
1	张*	男	35	经理
2	孙*	女	23	员工
3	王**	女	22	null
4	陈*	男	28	null

分开查询两张表的信息　查询

图 5.13　左外连接查询员工信息

说明： 在如图 5.13 所示的运行结果中，由于编号 3 与编号 4 的员工在部门表中没有对应的职位编号，因此在查询结果中以 null 的形式显示。

关键技术

外连接分为左外连接、右外连接、全外连接等。其中，左外连接的关键字为 LEFT JOIN，右外连接的关键字为 RIGHT JOIN。

下面举例说明内连接与外连接的区别。

假设有两张表，分别为表 A 与表 B，两张表公共的部分为 C。

内连接的连接结果是两个表都存在的记录，可以说 A 内连 B 得到的是 C。

表 A 左外连接 B，那么 A 不受影响，查询结果为公共部分 C 加表 A 的记录集。

表 A 右外连接 B，那么 B 不受影响，查询结果为公共部分 C 加表 B 的记录集。

全外连接表示两张表都不加限制。

设计过程

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等方法。具体代码参见配书光盘中的源程序。

（2）在页面中显示查询结果，具体代码如下：

```
<tr >
	<td width="20%" height="25" align="center">编号</td>
	<td width="20%" align="center">姓名</td>
	<td width="20%" align="center">性别</td>
	<td width="20%" align="center">年龄</td>
	<td width="20%" align="center">职位</td>
</tr>
<%
	String sql = "select e.id,e.name,e.sex,e.age,"+
	"d.deptname from tb_employee e left join tb_dept d on e.did=d.id";		//定义查询数据的 SQL 语句
	JDBCconnection db = new JDBCconnection();						//创建包含有操作数据库的类对象
	ResultSet rs = db.selectAll(sql);							//调用获取查询结果集的方法
	while(rs.next()){									//循环遍历查询结果集
%>
<tr align=center bgcolor="#f1f3f5">
	<td height="30" align="center"><%=rs.getString("id") %></td>			//在页面中显示查询结果
	<td><%=rs.getString("name") %></td>
	<td><%=rs.getString("sex") %></td>
	<td><%=rs.getString("age") %></td>
	<td><%=rs.getString("deptname") %></td>
</tr>
<%}%>
```

心法领悟 125：实现文件锁定。

文件锁定可以是独占，也可以是共享。可以使用 FileLock 类进行文件锁定。该类的主要方法有：isShared()方法，用于判断此锁定是否为共享的；isValid()方法，用于判断此锁定是否有效；release()方法，用于释放锁定。

实例 126 右外连接查询员工信息

光盘位置：光盘\MR\05\126

中级

实用指数：★★★

实例说明

在实例 125 中已经提到了右外连接，右外连接查询同样是为了保护数据的完整性。左外连接与右外连接的侧重点不同，开发人员可根据自己的需要选择适合的连接。本实例实现了部门信息表右外连接员工表，运行结果如图 5.14 所示。

右外连接查询员工信息

职位编号	职位名称	员工姓名	年龄
1	经理	张*	35
2	员工	孙*	23
null	null	王**	22
null	null	陈*	28

分开查询两张表的信息 查询

图 5.14 右外连接查询员工信息

关键技术

右外连接的运算符为 RIGHT JOIN。例如，表 A 右外连接表 B，结果为公共部分 C 加表 B 的结果集。如果表 A 中没有与表 B 匹配的项，就是用 NULL 进行连接。

设计过程

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等。具体代码参见配书光盘中的源程序。

（2）在页面中定义表格，显示查询结果。具体代码如下：

```
<tr >
	<td width="25%" height="25" align="center">职位编号</td>
	<td width="25%" align="center">职位名称</td>
	<td width="25%" align="center">员工姓名</td>
	<td width="25%" align="center">年龄</td>
</tr>
<%

	String sql = "select d.id,d.deptname,e.name,e.age "+
	"from tb_dept d right join tb_employee e on e.did=d.id";		//定义查询数据的 SQL 语句
	JDBCconnection db = new JDBCconnection();				//创建保存有查询数据的类对象
	ResultSet rs = db.selectAll(sql);					//调用获取查询结果集的方法
	while(rs.next()){							//循环遍历查询结果集
%>
<tr align=center bgcolor="#f1f3f5">
	<td height="30" align="center"><%=rs.getString("id") %></td>		//将查询结果显示在页面中
	<td><%=rs.getString("deptname") %></td>
	<td><%=rs.getString("name") %></td>
	<td><%=rs.getString("age") %></td>
</tr>
<%}%>
```

心法领悟 126：添加文字水印。

添加文字水印通常用于为图片添加本公司的网址或公司名称，从而标识本图片的所有权。应用 Java 中的绘图技术，可以实现对图片添加水印。Java 中提供的 Font 类封装了字体的大小、样式等属性。该类保存在 java.awt 包中，其构造方法可以指定字体的名称、大小和样式 3 个属性。具体语法如下：

```
Font (String name,int style,int size)
```

实例 127　多表外连接查询

光盘位置：光盘\MR\05\127

中级

实用指数：★★★

实例说明

外连接不仅可以用在两张表上，还可以将多个表进行外连接查询。本实例实现的是将员工表、职位表、学历表进行多表外连接查询，运行结果如图 5.15 所示。

说明：由于员工“陈*”只在员工表中存在，在其他两张表中并不存在，所以其他信息为空或者是默认值。

多表外连接查询员工信息

编号	姓名	性别	年龄	职位	学历	地址
1	张*	男	35	经理	本科	长春市绿园区
2	孙*	女	23	员工	高中	长春市二道区
3	王**	女	22	null	专科	长春市宽城区
4	陈*	男	28	null	null	null

分开查询三张表的信息　查询

图 5.15　多表外连接查询员工信息

关键技术

在本实例中两次使用了左外连接查询，即员工表与职位表进行左外连接后再与学历表进行左外连接，将职位表中符合条件的数据、学历表中符合条件的数据与员工表中的全部信息显示出来。若要给出外连接查询条件，应该使用 on 关键字。

设计过程

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等。具体代码参见配书光盘中的源程序。

（2）在页面中定义表格，显示查询结果。具体代码如下：

```
<table width="100%" height="80%" border="0">
    <tr >
        <td width="10%" height="25" align="center">编号</td>
        <td width="15%" align="center">姓名</td>
        <td width="10%" align="center">性别</td>
        <td width="10%" align="center">年龄</td>
        <td width="15%" align="center">职位</td>
        <td width="15%" align="center">学历</td>
        <td width="25%" align="center">地址</td>
    </tr>
        <%
            String sql = "select e.id,e.name,e.sex,e.age,d.deptname,a.degree,"+
            "a.address from (tb_employee e left join tb_dept d on"+
            " e.did=d.id)left join tb_empdata a on e.id=a.eid";                //定义查询 SQL 语句
            JDBCconnection db = new JDBCconnection();                          //定义操作数据库类对象
            ResultSet rs = db.selectAll(sql);                                  //调用查询方法
            while(rs.next()){                                                  //循环遍历查询结果集
        %>
    <tr     align=center bgcolor="#f1f3f5">
        <td width="10%" height="25" align="center"><%=rs.getInt("id") %></td>      //将查询结果显示在页面中
        <td width="15%" align="center"><%=rs.getString("name") %></td>
        <td width="15%" align="center"><%=rs.getString("sex") %></td>
        <td width="15%" align="center"><%=rs.getInt("age") %></td>
        <td width="15%" align="center"><%=rs.getString("deptname") %></td>
        <td width="15%" align="center"><%=rs.getString("degree") %></td>
        <td width="15%" align="center"><%=rs.getString("address") %></td>
    </tr>
    <%}%>
</table>
```

心法领悟 127：快速编写代码。

Eclipse 等开发工具提供了一些代码辅助功能，可以显著地提高编程速度。例如，在 Eclipse 的编辑区中输入

syso，之后按 Alt+/快捷键即可实现输入 System.out.println()。



实例 128　完全连接查询

光盘位置：光盘\MR\05\128

高级

实用指数：★★★★

实例说明

左外连接和右外连接都是针对某一张表的完整查询，如果要对连接的两张表都实现完整查询就要使用完全连接查询。完全连接查询的关键字为 FULL JOIN。本实例实现的是对部门表和员工表实现完全连接查询，运行结果如图 5.16 所示。

完全连接查询员工信息					
编号	部门名称	负责人	姓名	性别	学历
1	系统分析部	高岩	高岩	男	本科
2	软件测试部	刘芳	刘芳	女	研究生
3	软件测试部	刘芳	曹志	男	本科
4	设计部	蓝鹤	蓝鹤	女	研究生
5	开发部	李玉	李玉	男	专科
6	开发部	李玉	王洪	女	专科
7	设计部	蓝鹤	杨青	女	研究生
8	系统分析部	高岩	梅丽	女	本科
9	null	null	陈双	女	研究生

图 5.16　完全连接查询员工信息

关键技术

完全连接查询就是将左表的所有数据分别与右表的每条记录进行连接组合，返回的结果除了连接数据外，还有两个表中不符合条件的数据，并在左表或右表的相应列中填上 null 值。本实例中员工“陈双”只有员工表中存在，因此其他信息都是 null 值。

设计过程

在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等。接下来，在该类中定义将部门表和员工表进行完全连接查询的方法 getFull()，该方法以 List 集合作为返回值。具体代码如下：

```
public List getFull() {
    con = creatConnection();                                        //获取与数据库的连接
    ResultSet rest;
    List list = new ArrayList();
    try {
        Statement statement = con.createStatement();                //获取 Statement 对象
        String sql = "select e.id,dName,person,name,sex,schoolAge from tb_mrdept d full join tb_mremp e on d.id = e.dId";    //定义查询语句
        rest = statement.executeQuery(sql);                         //执行查询语句，获取查询结果集
        while (rest.next()) {
            MrEmp mrEmp = new MrEmp();                              //定义与数据表对应的 JavaBean 对象
            mrEmp.setId(rest.getInt(1));                            //设置对象属性
            mrEmp.setdName(rest.getString(2));
            mrEmp.setPerson(rest.getString(3));
            mrEmp.setName(rest.getString(4));
            mrEmp.setSex(rest.getString(5));
            mrEmp.setSchoolAge(rest.getString(6));
            list.add(mrEmp);
        }
        return list;                                                //返回查询结果
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

心法领悟 128：禁止网页被另存为。

在一些商务网站中，经常需要在网上发布一些重要信息，而有些重要信息只允许用户浏览，而不能进行下载或是将整个网页另存为。应用 JavaScript 脚本中的<noscript>标签即可防止网页被另存为——运行程序，进入网页进行浏览时，选择“文件”/“另存为”命令，在弹出的对话框中选择文件所要保存到的指定位置后，单击

“确定”按钮，此时将弹出一个提示框，显示“无法保存该网页”。<noscript>标签的语法如下：

```
<noscript>…</noscript>
```

5.3　嵌 套 查 询

实例 129　查询平均成绩在 85 分以上的学生信息

光盘位置：光盘\MR\05\129

中级

实用指数：★★★

实例说明

本实例实现的是查询学生信息表，将平均成绩在 85 分以上的学生信息在页面中显示出来，如图 5.17 所示。

学 生 信 息 查 询

当前位置>>>平均成绩高于85分的学生信息

选择查询条件：　所有学生信息　查询

ID	姓名	语文成绩	数学成绩	英语成绩
1	张*	98.0	97.0	100.0
3	李*	75.0	85.0	99.0

图 5.17　查询平均成绩高于 85 分的学生信息

关键技术

本实例实现的是查询学生信息表中的学生平均成绩。求平均成绩是指将语文成绩、数学成绩、英语成绩相加再除以 3 得到的结果，需要使用加号运算符进行内容相加后再进行除法运算，注意和聚集函数中的 AVG 函数区分开。

设计过程

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等。具体代码参见配书光盘中的源程序。

（2）创建名为 StuServlet 的 Servlet，在该 Servlet 中处理用户提交的请求，并作出相应的处理。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String tiaojian = request.getParameter("tiaojian");                          //获取用户提交内容
        if ("selectall".equals(tiaojian)) {                                          //判断用户提交内容
            request.getRequestDispatcher("index.jsp").forward(request,
                        response);                                                   //将地址转发至相应的页面
        }
        if ("selects".equals(tiaojian)) {
            String sql = "select StuID,StuName,Languages,Mathematics," +
                        "English from tb_student where (Languages+Mathematics+English)/3>85";  //定义查询 SQL 语句
            JDBCconnection db = new JDBCconnection();                                //创建包含有查询数据的类对象
            ResultSet rs = db.selectAll(sql);                                        //调用获取查询结果集方法
            request.setAttribute("rs", rs);                                          //将查询内容保存在 request 对象中
            request.getRequestDispatcher("stuSelect.jsp").forward(request,
                        response);                                                   //设置请求转发地址
        }
}
```

心法领悟 129：屏蔽 IE 主菜单。

可以通过 JavaScript 脚本中的 window 对象来屏蔽 IE 主菜单。应用 window 对象的 open()方法可以打开一个基于特定条件的新窗口，例如，可以指定新窗口的大小以及窗口中可用的选项，并且可以为引用它的窗口命名。

open()方法的语法如下：

```
window.open('StartExamfra.jsp','','width=790,height=430' menubar='no")
```

其中，参数 width 用来指定窗口的宽度；height 用来指定窗口的高度；menubar 用来指定菜单条，一般包括“文件”、“编辑”及其他一些菜单。

实例 130 多表统计本科学历部门经理的月收入情况

光盘位置：光盘\MR\05\130

中级

实用指数：★★★

实例说明

本实例实现的是查询员工表和职位表，将本科学历的部门经理的月收入情况查询出来，并显示在页面中，如图 5.18 所示。

员 工 信 息 查 询

当前位置>>统计本科部门经理的月收入

选择查询条件：所有员工及职位信息 查询

姓名	职位	部门	学历	工资
张*	经理	销售部	本科	3500.0
孙**	经理	技术部	本科	2700.0

图 5.18 本实例的运行结果

关键技术

本实例实现了嵌套查询。所谓嵌套查询，就是将多个查询语句组合在一起来实现一个功能。因此，掌握好单个查询语句的使用要点是很关键的，而灵活地运用好各种运算符也是很重要的。

设计过程

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等。具体代码参见配书光盘中的源程序。

（2）在项目中创建类 EmployeesDao，在该类中定义查询数据的方法 Select()，该方法以 ResultSet 对象作为返回值。具体代码如下：

```
public ResultSet Select() {
        JDBCconnection db = new JDBCconnection();                          //创建保存有操作数据库的类对象
        String sql = "select EmpName,JobTitle,Depatment,Degree,Wages" +
                    " from tb_employees,tb_position where Degree='本科' and " +
                    "Positions=(select PositionNumber from tb_position " +
                    "where JobTitle='经理') and Positions=PositionNumber";    //定义操作数据库的 SQL 语句
        ResultSet rs = db.selectAll(sql);                                    //调用查询方法，获取查询结果集
        return rs;
}
```

（3）在页面中显示查询结果，具体代码如下：

```
<tr>
 <td height="100" colspan="3" align="center"><table width="98%" height="80%" border="0">
   <tr>
  <td width="15%" height="25" align="center">姓名</td>
  <td width="15%" align="center">职位</td>
  <td width="27%" align="center">部门</td>
  <td width="26%" align="center">学历</td>
  <td width="27%" align="center">工资</td>
   </tr>
   <%
          ResultSet rs = (ResultSet)request.getAttribute("rs");          //获取保存在 request 对象中的集合内容
          while (rs.next()) {                                              //循环遍历查询结果
       %>
       <tr align=center bgcolor="#f1f3f5">
       <td height="30" align="center" bordercolor="#666666"><%=rs.getString("EmpName")%></td>
       <td><%=rs.getString("JobTitle")%></td>                             //将查询结果显示在页面上
       <td><%=rs.getString("Depatment")%></td>
       <td><%=rs.getString("Degree")%></td>
       <td><%=rs.getString("Wages")%></td>
       </tr>
<%}%>
```

心法领悟 130：使用 DISTINCT 关键字。

本实例中使用了 DISTINCT 关键字，在此要注意的是该关键字只能在 SELECT 列表中使用一次，并且不要在其后面使用逗号。DISTINCT 关键字最重要的一点，就是它并不是指某一行，而是指不重复 SELECT 输出的

所有列。



实例 131

在嵌套中使用 EXISTS 关键字

光盘位置：光盘\MR\05\131

高级

实用指数：★★★★☆

实例说明

EXISTS 关键字用于指定子查询中是否有指定的行存在。本实例在查询语句中嵌套了 EXISTS 关键字，实现查询商品表和商品登记表，并将已完成登记的商品信息显示在页面中，如图 5.19 所示。

已登记完商品信息	
商品名称	商品种类
可乐	饮料
香肠	食品
分开查询两张表的信息	查询

图 5.19　在嵌套中使用 EXISTS 关键字查询已登记完的商品信息

关键技术

如果要在子查询中返回一个真值或假值，只考虑是否满足谓词条件，数据内容本身并不重要。如果子查询返回一行或多行，则返回 true，否则返回 false。要使 EXISTS 谓词有用，应该在查询中建立查询条件以匹配连接建立起来的两个表中的值。

语法如下：

```
EXISTS subquery
```

参数说明

subquery：不允许有 COMPUTE 子句和 INTO 关键字的一个受限 SQL 语句。

其结果类型为 Boolean，如果子查询包含行，则返回 true，否则返回 flase。

设计过程

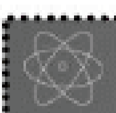
（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等。具体代码参见配书光盘中的源程序。

（2）在页面中显示查询结果，具体代码如下：

```
<table width="100%" height="80%" border="0">
        <tr >
                <td width="50%" height="25" align="center">商品名称</td>
        <td width="50%" align="center">商品种类</td>
        </tr>
        <%
                String sql = "select a.name,a.kind from tb_comm a where"+
                " exists (select * from tb_comminfo b where b.cid=a.id)";
                JDBCconnection db = new JDBCconnection();
                ResultSet rs = db.selectAll(sql);
                while(rs.next()){
        %>
        <tr align=center bgcolor="#f1f3f5">
                <td height="30" align="center"><%=rs.getString("name") %></td>
                <td><%=rs.getString("kind") %></td>
        </tr>
<%}   %>
```

心法领悟 131：获取分页的总页数。

分页技术在 Web 中应用得十分广泛。在分页过程中，计算分页后的总页数十分关键。总页数可以根据公式“总记录数/每页显示的记录数”来计算，但需要注意一点，如果“总记录数/每页显示的记录数”有余数，则需要将总页数加 1。然而，在程序编写时，无法准确地判断“总记录数/每页显示的记录数”是否有余数。此时可以将该公式修改为“总页数=(总记录数−1)/每页显示的记录数+1”，这样无论在“总记录数/每页显示的记录数”是否整除的情况下，都可以准确地计算出分页后的总页数。



实例 132 动态指定查询条件

光盘位置：光盘\MR\05\132

中级

实用指数：★★★

实例说明

动态指定查询条件，是指用户可以选择查询的条件和查询的内容。本实例实现的是动态查询学生信息表中的内容，用户可选择按 id 查询或按姓名查询，并可以手动添加查询条件。本实例的运行结果如图 5.20 所示。

动态指定查询条件

条件：按id查询 [查询]

ID	姓名	班级	籍贯	专业
1	张**	三年四班	长春市	物理
2	李*	一年五班	北京市	语文
3	王*	二年六班	沈阳市	数学

图 5.20　动态指定查询条件

关键技术

本实例实现了在 Servlet 中处理用户提交的请求。获取前台页面中提交的内容可以使用 request 对象的 getParameter()方法，但是需要注意的是，一定要使用过滤器进行处理，否则如果在前台页面中输入中文，就会出现中文乱码问题，得不到想要的结果。

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等。具体代码参见配书光盘中的源程序。

（2）创建名为 StuServlet 的 Servlet，在该 Servlet 中判断用户提交的请求，并作出相应的处理。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String tiaojian = request.getParameter("tiaojian");                    //获取用户提交的查询条件
        String sou=request.getParameter("sou");                                //获取页面中请求的内容
        String sql="";
        JDBCconnection db=new JDBCconnection();                                //创建保存有操作数据库的类对象
        if ("selectid".equals(tiaojian)) {                                     //判断用户的查询条件
                sql="select * from tb_stu where id="+sou;                      //定义查询 SQL 语句
                ResultSet res=db.selectAll(sql);                               //调用查询方法，获取查询结果集
                request.setAttribute("rs", res);                               //将查询结果集保存在 request 对象中
                request.getRequestDispatcher("stuSelect.jsp").forward(request, response);  //设置请求转发地址
        }
        if ("selectname".equals(tiaojian)) {
                sql="select * from tb_stu where name like '%"+sou+"%'";        //定义模糊查询 SQL 语句
                ResultSet res=db.selectAll(sql);                               //调用查询结果，获取查询结果集
                request.setAttribute("rs", res);
                request.getRequestDispatcher("stuSelect.jsp").forward(request, response);
        }
}
```

（3）在页面中显示查询结果，具体代码如下：

```
<table width="98%" height="80%" border="0">
        <tr>
                <td width="20%" height="25" align="center">ID</td>
                <td width="20%" align="center">姓名    </td>
                <td width="20%" align="center">班级    </td>
                <td width="20%" align="center">籍贯    </td>
                <td width="20%" align="center">专业     </td>
        </tr>
        <%
                ResultSet rs =(ResultSet)request.getAttribute("rs");           //获取保存在 request 对象中的查询结果集
                while (rs.next()) {                                            //循环遍历查询结果集
        %>
        <tr align=center bgcolor="#f1f3f5">
                <td height="30" align="center"><%=rs.getString("id")%></td>
                <td><%=rs.getString("name") %></td>                            //将查询结果显示在页面中
                <td><%=rs.getString("class") %></td>
```

```
                <td><%=rs.getString("native")%></td>
                <td><%=rs.getString("professional")%></td>
        </tr>
        <%}  %>
</table>
```

秘笈心法

心法领悟 132：在实际开发中，当存在以下情况时可以将表中列的值设置为 null。

- 其值未知。例如，学生表中的班级列，在刚入学时可能将其值设置为 null。
- 其值不存在。
- 数据表中列值不可用。例如，雇员表中包含一个经理编号列，可将不是经理的员工的经理编号一列设置为 null。

5.4　常见谓词的使用

实例 133　应用 PATINDEX 谓词进行模糊查询

光盘位置：光盘\MR\05\133

中级　实用指数：★★★

实例说明

除了应用 LIKE 关键字进行模糊查询，还有一种可以进行模糊查询的方法——patindex()函数。该函数可以返回指定字符串在表达式中第一次出现的位置。本实例实现的是查询员工信息表，并对姓名进行模糊查询，运行结果如图 5.21 所示。

员　工　信　息　查　询

当前位置>>>所有员工信息

按姓名模糊搜索：　　搜索

姓名	档案号	部门	职位	工资
王一	1005	质量部	主任	1600.4
王五	1011	技术部	主任	1500.1

图 5.21　按姓名进行模糊查询

关键技术

patindex()函数主要用于搜索字符或字符串。如果被搜索的字符串中包含要搜索的字符，那么该函数返回一个非零的整数，表示 pattern 字符串在表达式 expression 中第一次出现的位置，起始值为 1。patindex()函数支持使用通配符来进行搜索。

语法如下：

```
patindex('%pattern%',expression)
```

参数说明

❶ pattern：要搜索的字符串，可以使用通配符。pattern 之前和之后需要使用“%”，除非搜索的字符串在被搜索的字符串的最前面或最后面。

❷ expression：表达式，表示被搜索的字符串。expression 通常是一个表中的字段。

该函数中还可以使用“[ ]”来返回某些指定字符中之一的位置。例如：

```
select patindex('%[c,d]%','abcdeft')
```

上述代码的含义是搜索字符“c”或“d”中的一个在表达式“abcdeft”中最先出现的位置。由于“c”在字符串中第一次出现的位置是 3，因此该句代码返回值为 3。

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等。具体代码参见配书光盘中的源程序。

（2）在项目中定义类 WorkDao，在该类中定义查询方法 selects()，该方法以 ResultSet 对象作为返回值。具体代码如下：

```
public ResultSet selects(String tiaojian) {
        JDBCconnection db = new JDBCconnection();                    //定义保存有查询数据库的类对象
```

```
        String sql = "";
        if (tiaojian == "" || tiaojian == null) {                               //判断用户提交内容是否为空
            sql = "select EmpName,Filenumber,Depatment,Positions,Wages from tb_works";        //如果没有查询条件，查询表中所有值
        } else {
            sql = "select EmpName,Filenumber,Depatment,Positions,Wages from tb_works where patindex('%" + tiaojian
                    + "%',EmpName)>0";                                          //定义查询 SQL 语句
        }
        ResultSet rs = db.selectAll(sql);                                      //调用查询数据库方法
        return rs;                                                             //返回查询结果集
}
```

秘笈心法

心法领悟 133：关键字和保留字。

很多人认为保留字就是关键字，其实严格地讲，关键字和保留字是不同的，但是一般的程序语言并没有严格地区分它们。简单地说，关键字是指程序中一定会用到的单词，而保留字是指系统保留起来不给用户使用，但是自己也不会用到的单词，如 goto 就属于保留字而不属于关键字。

实例 134 在查询中使用四舍五入谓词 ROUND

光盘位置：光盘\MR\05\134

中级

实用指数：★★★

实例说明

要实现对数据表中的数据进行四舍五入，可以使用 ROUND 函数，该函数将返回数字表达式并四舍五入为指定的长度或精度。本实例实现将员工信息表中的员工工资进行四舍五入，运行结果如图 5.22 所示。

员 工 信 息 查 询

当前位置>>>所有员工四舍五入后工资情况

选择查询条件： 所有员工正常工资 查询

姓名	档案号	部门	职位	工资
张三	1012	技术部	员工	1650.0
孙**	1004	销售部	经理	2501.0
王一	1005	质量部	主任	1600.0
小*	1007	质量部	经理	1701.0
李四	1010	技术部	经理	2001.0
王五	1011	技术部	主任	1500.0

图 5.22 将员工工资进行四舍五入

关键技术

四舍五入函数 ROUND 的具体语法如下：

```
ROUND(expression,length)
```

参数说明

❶ expression：精确数字或近似数据类型类别的表达式。

❷ length：要四舍五入的精度。该参数必须是 tinyint、smallint 或 int 类型。

该方法的返回值与参数 expression 的类型相同。

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等。具体代码参见配书光盘中的源程序。

（2）本实例实现两项功能，分别为查询所有员工工资、查询四舍五入后的员工工资。创建名为 WorkServlet 的 Servlet，在该 Servlet 中对用户提交的请求作出相应的处理。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
```

```
        String tiaojian = request.getParameter("tiaojian");                          //获取用户提交内容
        if ("selectall".equals(tiaojian)) {                                          //判断用户提交的内容
                request.getRequestDispatcher("index.jsp")                            //将请求转发至 index.jsp 页面
                                .forward(request, response);
        }
        if ("selects".equals(tiaojian)) {
                String sql = "select EmpName,FileNumber,Depatment,Positions,round(Wages,0) Wages from tb_works";        //定义查询 SQL 语句
                JDBCconnection db = new JDBCconnection();                            //定义包含有操作数据库的类对象
                ResultSet rs = db.selectAll(sql);                                    //调用查询数据方法
                request.setAttribute("rs", rs);                                      //将查询结果保存在 request 对象中
                request.getRequestDispatcher("workRound.jsp").forward(request, response);    //将请求转发至 index.jsp 页面
        }
}
```

（3）在页面中定义表格，显示查询结果。具体代码如下：

```
<td height="100" colspan="3" align="center"><table width="98%" height="80%" border="0">
  <tr>
        <td width="15%" height="25" align="center">姓名</td>
        <td width="15%" align="center">档案号</td>
        <td width="27%" align="center">部门</td>
        <td width="26%" align="center">职位</td>
        <td width="27%" align="center">工资</td>
  </tr>
  <%
        ResultSet rs = (ResultSet)request.getAttribute("rs");                    //获取保存在 request 对象中的查询结果集
                while (rs.next()) {                                               //循环遍历查询结果集
   %>
  <tr align=center bgcolor="#f1f3f5">
        <td height="30" align="center"><%=rs.getString("EmpName")%></td>
        <td><%=rs.getString("FileNumber")%></td>                                  //将查询结果显示在页面中
        <td><%=rs.getString("Depatment")%></td>
        <td><%=rs.getString("Positions")%></td>
        <td><%=rs.getString("Wages")%></td>
  </tr>
  <%}%>
```

■ 秘笈心法

心法领悟 134：用于匹配汉字的正则表达式。

正则表达式中提供了众多的文字字符、元字符（如“+”、“$”）、字符集（如“\d”），使用这些字符可以完成想要匹配的内容。但是如果需要将一段文字中的汉字查询出来，使用这些字符就无法完成匹配了。此时需要使用汉字的 Unicode 编码来进行匹配，如使用表达式[\u4e00-\u9fa5]来判断是否为中文。

实例 135　查询比质量部所有员工工资都高的员工信息

光盘位置：光盘\MR\05\135

高级

实用指数：★★★★

■ 实例说明

实现查询比质量部所有员工工资都高的员工信息，首先要获取质量部中最高的员工工资，之后再进行查询，因此需要使用子查询。本实例的运行结果如图 5.23 所示。

员　工　信　息　查　询

当前位置>>>比质量部所有员工工资都高的员工信息

选择查询条件：所有员工正常工资　查询

姓名	档案号	部门	职位	工资
孙**	1004	销售部	经理	4500.5
李四	1010	技术部	经理	5000.7
王五	1011	技术部	主任	4500.1

图 5.23　查询比质量部所有员工工资都高的员工信息

■

要实现本实例，需要将员工的工资与质量部最高的员工工资进行比较，如果高于质量部最高的工资，则返回查询结果。获取质量部最高的工资使用的是 MAX 函数。

设计过程

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等。具体代码参见配书光盘中的源程序。

（2）本实例实现两项功能，分别为查询所有员工的工资信息、查询高于质量部所有员工工资的员工信息。创建名为 WorkServlet 的 Servlet，在该 Servlet 中处理用户请求，并作出相应的处理。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    String tiaojian = request.getParameter("tiaojian");                    //获取用户提交的请求信息
    if ("selectall".equals(tiaojian)) {                                     //判断用户提交的请求
        request.getRequestDispatcher("index.jsp")
                .forward(request, response);                                //设置请求转发地址
    }
    if ("selects".equals(tiaojian)) {
        String sql = "select EmpName,FileNumber,Depatment,Positions,Wages from tb_works" +
                " where Wages>(select max(Wages) from tb_works where Depatment='质量部')"; //定义查询 SQL 语句
        JDBCconnection db = new JDBCconnection();                           //定义保存有操作数据库的类对象
        ResultSet rs = db.selectAll(sql);                                   //调用查询数据方法
        request.setAttribute("rs", rs);                                     //将查询结果保存在 request 对象中
        request.getRequestDispatcher("workSelect.jsp").forward(request,
                response);
    }
}
```

秘笈心法

心法领悟 135：使用 equals()方法时的注意事项。

在比较两个对象时，使用了 equals()方法。未重载 equals()方法的类的对象使用该方法与另一个对象进行比较时，将返回 false，即使这两个对象拥有相同的内容。这是因为这个未重载 equals()方法的类实际上调用了 Object 类的 equals()方法。

实例 136 查询工资高于质量部任意一名员工的员工信息

光盘位置：光盘\MR\05\136

中级　实用指数：★★★

实例说明

本实例将实例 135 进行了修改，实现查询工资高于质量部任意一名员工的员工信息，运行结果如图 5.24 所示。

图 5.24　查询工资高于质量部任意一名员工的员工信息

关键技术

要实现本实例，需要获取质量部最低的员工工资，再将员工表中的工资情况与质量部最低的员工工资进行比较，如果大于最低的员工工资，则返回查询内容。

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等。具体代码参见光盘中的源程序。

（2）本实例实现两项功能，一是查询所有员工工资信息；二是查询工资高于质量部任意一名员工的员工信息。创建名为 WorkServlet 的 Servlet，在该 Servlet 中获取用户提交请求，并作出相应的处理。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    String tiaojian = request.getParameter("tiaojian");             //获取用户提交内容
    if ("selectall".equals(tiaojian)) {                              //判断用户提交的请求内容
```

```
            request.getRequestDispatcher("index.jsp")                    //设置请求转发地址
                    .forward(request, response);
        }
        if ("selects".equals(tiaojian)) {                                //判断用户提交的请求内容
            String sql = "select EmpName,FileNumber,Depatment,Positions,Wages from tb_works where Wages>" +
                    "(select min(Wages) from tb_works where Depatment='质量部') and Depatment!='质量部'";    //定义查询 SQL 语句
            JDBCconn db = new JDBCconn();                                //创建保存有查询数据的类对象
            ResultSet rs = db.select(sql);                               //调用查询方法获取查询结果集
            request.setAttribute("rs", rs);                              //将查询结果集保存在 request 对象
            request.getRequestDispatcher("workSelect.jsp").forward(request,
                    response);                                           //将请求转发至相应的页面
        }
}
```

（3）在页面中定义表格，显示查询结果。具体代码如下：

```
<td height="100" colspan="3" align="center"><table width="98%" height="80%" border="0">
   <tr>
          <td width="15%" height="25" align="center">姓名</td>
          <td width="15%" align="center">档案号</td>
          <td width="27%" align="center">部门</td>
          <td width="26%" align="center">职位</td>
       <td width="27%" align="center">工资</td>
   </tr>
   <%
            ResultSet rs = (ResultSet)request.getAttribute("rs");
            while (rs.next()) {
      %>
   <tr align=center bgcolor="#f1f3f5">
         <td height="30" align="center"><%=rs.getString("EmpName")%></td>
            <td><%=rs.getString("FileNumber")%></td>
            <td><%=rs.getString("Depatment")%></td>
            <td><%=rs.getString("Positions")%></td>
            <td><%=rs.getString("Wages")%></td>
   </tr>
  <%}%>
```

秘笈心法

心法领悟 136：解析 Eclipse 报错的原因。

有些读者下载完 Eclipse 时，可能会出现无法打开的错误。可以通过以下方法来解决：（1）如果没有安装 JDK，在本机上安装 JDK 即可；（2）在 Eclipse 的 Preferences 中重新设置 JDK 路径；（3）将 Eclipse 目录下的 configutation 文件夹中的文件，除 config.ini 文件外，全部删除。之后重新启动 Eclipse 即可。

实例 137　应用 UNION 谓词消除重复的行

光盘位置：光盘\MR\05\137

中级

实用指数：★★★

实例说明

去除重复值查询是一种很常见的查询方式，实现方式有很多，如本例将要介绍的使用 UNION 关键字实现消除重复的行。本实例将查询学生信息表中的信息，并将学生来自的城市信息显示在页面中，如图 5.25 所示。

图 5.25　本实例的运行结果

将两个或更多查询的结果组合为单个结果集，该结果集包含联合查询中的所有查询的全部行。这与使用连接组合两个表中的列不同。使用 UNION 组合两个查询的结果集的基本规则如下：

- 所有查询中的列数和列的顺序必须相同。
- 数据类型必须兼容。

语法如下：

```
{ < query specification > | ( < query expression > ) }
        UNION [ ALL ]
< query specification | ( < query expression > )
        [ UNION [ ALL ] < query specification | ( < query expression > )
[ ...n ] ]
```

参数说明

< query specification >：查询规范或查询表达式，用以返回与另一个查询规范或查询表达式所返回的数据组合的数据。

设计过程

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等。具体代码参见配书光盘中的源程序。

（2）本实例实现两项功能，一是实现查询所有学生信息；二是实现查询学生的来源地。创建名为 StuServlet 的 Servlet，在该 Servlet 中处理用户提交的请求并作出相应的处理。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String tiaojian=request.getParameter("tiaojian");                          //获取用户提交的请求信息
        if("selects".equals(tiaojian)){                                            //判断用户提交的请求信息
            JDBCconnection db=new JDBCconnection();                                //创建保存有操作数据库的类对象
            String sql="select Native from tb_stuone union select Native from tb_stutwo";   //定义查询 SQL 语句
            ResultSet res=db.selectAll(sql);                                       //调用查询方法
            request.setAttribute("rs", res);                                       //将查询结果保存在 request 对象中
            request.getRequestDispatcher("stuSelect.jsp").forward(request, response);   //设置请求转发地址
        }
        if("selectall".equals(tiaojian)){
            request.getRequestDispatcher("index.jsp").forward(request, response);
        }
}
```

秘笈心法

心法领悟 137：聚合函数使用注意事项。

当使用 CUBE 或 ROLLUP 时，不支持区分聚合，例如，AVG(DISTINCT column_name)、COUNT(DISTINCT column_name)、MAX(DISTINCT column_name)、MIN(DISTINCT column_name)和 SUM(DISTINCT column_ name)。如果使用了，Microsoft® SQL Server™ 将返回错误信息并取消查询。

实例 138	应用 UNION ALL 谓词保留重复行 光盘位置：光盘\MR\05\138	高级 实用指数：★★★★

实例说明

UNION 关键字与 ALL 关键字组合使用，可以保留查询结果中的重复值。在此需要注意的是，使用 UNION ALL 组合查询的两张表必须要有相同的结构。本实例实现的是查询 tb_stuone 表与 tb_stutwo 表，将学生姓名、班级、籍贯显示在页面中，如图 5.26 所示。

学 生 信 息 查 询

当前位置>>>所有学生信息

选择查询条件： 各班学生信息 查询

姓名	班级	籍贯
王*强	二年一班	长春
李*强	二年一班	吉林
孙*国	二年一班	沈阳
李*	二年二班	沈阳
王*	二年二班	长春
孙*	二年二班	吉林

图 5.26 使用 UNION ALL 组合查询学生信息

当组合使用 UNION 和 ALL 关键字时，不删除重复行，也不对行自动排序。这种模式需要的计算资源少，所以应尽可能使用，尤其是处理大型表时。例如，下列情况就应该使用 UNION ALL 组合查询。

- ❑ 知道有重复行并想保留这些行。
- ❑ 知道不可能有任何重复的行。
- ❑ 不考虑是否有任何重复的行。

设计过程

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等。具体代码参见配书光盘中的源程序。

（2）本实例实现两项功能，一是将 tb_stuone 表与 tb_stutwo 表分开进行查询；二是使用 UNION ALL 关键字进行组合查询。创建名为 StuServlet 的 Servlet，在该 Servlet 中处理用户提交的请求，并作出相应的处理。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String tiaojian=request.getParameter("tiaojian");                    //获取用户提交内容
        if("selects".equals(tiaojian)){                                      //判断用户提交的请求信息
            JDBCconnection db=new JDBCconnection();                          //创建保存有操作数据库的类对象
            String sql="select StuName,Class,Native from tb_stuone union all select StuName,Class,Native from tb_stutwo";//定义查询 SQL 语句
            ResultSet res=db.selectAll(sql);                                 //调用查询方法获取查询结果集
            request.setAttribute("rs", res);                                 //将查询结果保存在 request 对象内
            request.getRequestDispatcher("stuSelect.jsp").forward(request, response);  //定义请求转发地址
        }
        if("selectall".equals(tiaojian)){
            request.getRequestDispatcher("index.jsp").forward(request, response);
        }
}
```

（3）在页面中定义表格，显示查询结果。具体代码如下：

```
<tr>
    <td height="100" colspan="3" align="center"><table width="98%" height="80%" border="0">
        <tr >
            <td height="30" align="center">姓名</td>
            <td align="center">班级</td>
            <td align="center">籍贯</td>
        </tr>
    <%
      ResultSet rs=(ResultSet)request.getAttribute("rs");                    //获取保存在 request 对象中的信息
       while(rs.next())                                                      //循环遍历查询结果集
            {
     %>
<tr align=center bgcolor="#f1f3f5">
    <td height="30"><%=rs.getString("StuName")%></td>                       //将查询出的信息显示在页面中
    <td><%=rs.getString("Class")%></td>
    <td><%=rs.getString("Native")%></td>
</tr>
<%} %>
```

秘笈心法

心法领悟 138：Servlet 的线程安全问题。

Servlet 采用多线程模式运行，并为并发的每个访问请求都预备一个独立的线程进行响应。这种模式可以提高访问性能，但也带来了线程安全的问题。例如，如果并发的多个请求访问的是同一个 Servlet，那么将有多个线程来并发调用该 Servlet 的 service()方法。

实例 139　查询各商品销售额所占的百分比

高级

光盘位置：光盘\MR\05\139

实用指数：★★★★

实例说明

很多程序都要求将一些数据以百分比的形式显示出来，以方便用户查询。本实例实现的是查询商品表中的

各商品销售额占总销售额的百分比，运行结果如图 5.27 所示。

商 品 信 息 查 询

当前位置>>>商品销售金额的百分比

选择查询条件： 商品销售信息 查询

商品名称	销售百分比
香水	18.18%
熟食	27.27%
酒水	54.55%

图 5.27　本实例的运行结果

关键技术

要实现本实例，首先需要利用 SUM 函数将总的销售额计算出来，再将某商品的销售额÷总的销售额×100%，即可得到该商品销售额所占的百分比。

设计过程

（1）在项目中创建类 JDBCconnection，在该类中定义操作数据库的相关方法，其中包括获取数据库连接、关闭数据库连接、获取查询结果集等。具体代码参见配书光盘中的源程序。

（2）本实例实现两项功能，分别为查询商品的总销售额和各商品销售额占总销售额的百分比。创建名为 CommodityServlet 的 Servlet，在该 Servlet 中处理用户提交的请求，并作出相应的处理。具体代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    String tiaojian=request.getParameter("tiaojian");                       //获取用户提交的请求内容
    if("selects".equals(tiaojian)){                                          //判断用户提交的请求内容
        JDBCconnection db=new JDBCconnection();                              //创建保存有操作数据库的类对象
        String sql="select CommName,round(Total/(select sum(Total)" +
                " from tb_commodity)*100,2) Percentage from tb_commodity";   //定义查询语句
        ResultSet res=db.selectAll(sql);                                     //调用查询数据方法
        request.setAttribute("rs", res);                                     //将查询结果集保存在 request 对象中
        request.getRequestDispatcher("commodity.jsp").forward(request, response);   //定义请求转发地址
    }
    if("selectall".equals(tiaojian)){
        request.getRequestDispatcher("index.jsp").forward(request, response);
    }
}
```

（3）在页面中定义表格，显示查询内容。具体代码如下：

```
<table width="100%" border="0">
    <tr>
        <td width="50%" height="25" align="center">商品名称</td>
        <td align="center">销售百分比</td>
    </tr>
    <%
        ResultSet rs=(ResultSet)request.getAttribute("rs");                  //获取保存在 request 对象中的内容
        while(rs.next()){                                                    //循环遍历查询结果集
    %>
    <tr align=center bgcolor="#f1f3f5">
        <td height="30" align="center">
            <%=rs.getString("CommName") %>                                   //将查询结果显示在页面中
        </td>
        <td>
            <%=rs.getString("Percentage") %>%
        </td>
    </tr>
    <%} %>
```

心法领悟 139：区分 Servlet 的编译和运行环境。

编译和运行 Servlet 的环境是完全不同的两个环境。解释执行 Servlet 的过程是在 Tomcat 服务器程序中进行的，Servlet 的运行环境就是 Tomcat 服务器所设置的环境；编译 Servlet 是由编程人员使用 javac 命令完成的，Servlet 的编译环境是某个命令行窗口或开发工具所设置的环境。

第 6 章

数据库高级应用

- 在 Java Web 程序中调用存储过程
- 使用触发器
- 使用批处理
- 使用视图

6.1 在 Java Web 程序中调用存储过程

实例 140 调用存储过程实现用户身份的验证 高级

光盘位置：光盘\MR\06\140 实用指数：★★★★

实例说明

存储过程是为完成特定的功能而汇集在一起的一组经编译后存储在数据库中的 SQL 语句。存储过程可以用参数的形式输入、输出数据，并支持用户设计的变量和流程控制。一个存储过程中可以包含查询、插入、删除、更新等操作，当一个存储过程被执行时，这些操作也会同时执行。本实例实现的是用户登录时，调用存储过程来验证用户的身份，运行结果如图 6.1 所示。



图 6.1 调用存储过程实现用户身份的验证

关键技术

要实现本实例，首先要在 SQL Server 2000 数据库下创建存储过程，之后在 Java 程序中调用存储过程。主要用到 SQL 语句中的 CREATE PROCEDURE 来创建存储过程，其语法如下：

```
CREATE PROC [ EDURE ] procedure_name [ ; number ]
    [ { @parameter data_type }
        [ VARYING ] [ = default ] [ OUTPUT ]
    ] [ ,...n ]
[ WITH
    { RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION } ]
[ FOR REPLICATION ]
AS sql_statement [ ...n ]
```

参数说明

❶ procedure_name：表示新存储过程的名称。

❷ number：表示可选的整数，用来对同名的存储过程分组，以便用一条 DROP PROCEDURE 语句将同组的存储过程一起删除。

❸ @parameter：表示存储过程中的参数。

❹ data_type：表示参数的数据类型。

在 Java 程序中调用存储过程来实现登录验证。调用存储过程的语法如下：

```
{ CALL procname (?,?)}
```

参数说明

❶ procname：表示存储过程的名称。

❷ ?：表示传递的参数。

（1）在数据库中创建存储过程 validateSelect，该存储过程保存有两个字符类型的参数，分别代表用户名与密码，并按照这两个参数查询数据表。具体代码如下：

```
create procedure validateSelect
@userName varchar(20),
@password varchar(20)
as select * from tb_user where userName = @userName and password = @password
```

（2）在项目中创建类 TransferProcure，在该类中定义调用存储过程的方法 executeQuery()（该方法包含两个 String 类型的参数）。具体代码如下：

```
public String executeQuery(String userName,String passWord){
    String message = "验证失败";                                    //定义保存返回值的字符串对象
```

```
        conn = getConn();                                           //获取数据库连接
        CallableStatement cs = null;                                //定义 CallableStatement 对象
        String sql = "{call validateSelect('"+userName+"','"+passWord+"')}";  //定义调用存储过程语句
        try {
            cs = conn.prepareCall(sql);                             //调用存储过程
            ResultSet rest = cs.executeQuery();                     //获取结果集
            while(rest.next()){                                     //循环遍历结果集对象
                message = "验证成功";                                //设置对象信息
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return message;                                             //返回 String 对象
}
```

秘笈心法

心法领悟 140：系统存储过程。

系统存储过程存储在 master 数据库中，并以“sp_”为前缀，主要用来从系统表获取信息，为系统管理员管理 SQL Server 提供帮助，为用户查询数据库对象提供方便。例如，用来查看数据库对象信息的系统存储过程 sp_help，显示存储过程和其他对象的文本存储过程 sp_helptext 等。

实例 141	调用存储过程添加数据 光盘位置：光盘\MR\06\141	高级 实用指数：★★★★

实例说明

在实际开发中，经常需要对数据进行操作。由于存储过程是线程同步的，并且具有事务回滚的功能，因此应用存储过程对数据库进行操作比较安全。本实例实现的是调用存储过程来添加数据，运行结果如图 6.2 所示。

图 6.2　调用存储过程添加数据

关键技术

在 Java 中可以使用 CallableStatement 接口来创建调用存储过程的语句。示例代码如下：

```
CallableStatement cs = con.prepareCall(sql);
```

此语句将创建一个 CallableStatement 对象，使用该对象即可完成向存储过程传递参数的功能。示例代码如下：

```
cs.setString(1,"mm");
```

在此语句中，1 代表存储过程中参数的序列位置，mm 代表所传递的参数。

执行存储过程可以使用 executeUpdate 语句。示例代码如下：

```
cs.executeUpdate();
```

（1）在数据库中创建存储过程，当用户调用该存储过程时，实现向用户表中添加数据。代码如下：

```
create procedure insertUser
@userName varchar(20),
```

```
@passWord varchar(20),
@age int,
@sex varchar(4),
@job varchar(20)
as
insert into tb_user values(@userName,@passWord,@age,@sex,@job)
go
```

（2）在项目中创建类 UserUtil，在该类中定义操作数据库的相关方法，然后定义调用存储过程的方法 executeUpdate()，该方法以与数据表对应的 JavaBean 对象为参数。具体代码如下：

```
public boolean executeUpdate(User user) {
    conn = getConn();                                          //获取数据库连接
    CallableStatement cs = null;                               //定义 CallableStatement 对象
    String sql = "{call insertUser('" + user.getUserName() + "','"
            + user.getPassWord() + "','" + user.getAge() + "','"
            + user.getSex() + "','" + user.getJob() + "')}";   //定义调用存储过程的 SQL 语句
    try {
        cs = conn.prepareCall(sql);                            //实例化 CallableStatement 对象
        cs.executeUpdate();                                    //执行 SQL 语句
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}
```

秘笈心法

心法领悟 141：创建存储过程的注意事项。

存储过程的最大大小为 128MB。用户定义的存储过程只能在当前数据库中创建（临时存储过程除外，临时存储过程总是在 tmppdb 中创建）。在单个批处理中，CREATE PROCEDURE 语句不能与其他 Transact-SQL 语句组合使用。

实例 142　调用加密存储过程

光盘位置：光盘\MR\06\142

高级

实用指数：★★★★

实例说明

有时为了程序的安全，会将存储过程进行加密。这样用户在执行 exec sp_helptext 命令后，将无法查看存储过程的源代码，但是通过 Java 程序还是可以正常地调用存储过程的。本实例首先在数据库中创建加密的存储过程，实现查询用户表中的信息，然后在 Java 程序中调用该存储过程，在页面中显示相应信息，如图 6.3 所示。

调用存储过程>>>

查询数据表中全部信息

编号	用户名	年龄	性别	工作
1	mr	30	女	程序员
2	绿草	26	男	程序员
3	春意盎然	30	男	副总
4	小双	25	女	客服

图 6.3　调用加密存储过程查询数据表中全部信息

调用加密的存储过程和调用普通的存储过程的方式是相同的，在此重点介绍的是如何创建加密的存储过程，其语法如下：

```
CREATE PROCEDURE proc_name
WITH ENCRYTION
AS
sql_statement
```

参数说明

❶ proc_name：存储过程名称。

❷ WITH ENCRYTION：加密存储过程的关键字。

❸ sql_statement：存储过程中的一个或多个 SQL 语句。

设计过程

（1）在数据库中创建加密存储过程，用于查询用户表中所有数据。具体代码如下：

```
create procedure selectUser
with encryption
as
   select * from tb_user
go
```

（2）在项目中创建类 UserUtil，在该类中定义操作数据库的各种方法，然后定义调用存储过程的方法 executeUpdate()，该方法返回值为 List 集合对象。具体代码如下：

```
public List executeUpdate() {
      conn = getConn();                          //获取数据库连接
      CallableStatement cs = null;               //定义 CallableStatement 对象
      String sql = "{call selectUser}";          //定义调用存储过程的 SQL 语句
      List list = new ArrayList();
      try {
            cs = conn.prepareCall(sql);          //实例化 CallableStatement 对象
            ResultSet rest = cs.executeQuery();  //执行 SQL 语句
            while(rest.next()){                  //循环遍历查询结果集
                  User user = new User();        //定义与数据表对应的 JavaBean 对象
                  user.setId(rest.getInt(1));    //设置对象属性
                  user.setUserName(rest.getString(2));
                  user.setAge(rest.getInt(4));
                  user.setSex(rest.getString(5));
                  user.setJob(rest.getString(6));
                  list.add(user);                //向集合中添加对象
            }
      } catch (SQLException e) {
            e.printStackTrace();

      }
      return list;
}
```

秘笈心法

心法领悟 142：不要使用 sp_prefix 作为存储过程名称。

sp_prefix 是系统存储过程保留的，数据库引擎将始终首先在数据库中查找具有此前缀的存储过程。这意味着引擎首先检查主数据库，然后检查存储过程实际所在的数据库，将需要较长的时间才能完成检查过程。而且，如果碰巧存在一个名称相同的系统存储过程，则程序员创建的数据库根本不会得到处理。

实例 143　获取数据库中所有存储过程

光盘位置：光盘\MR\06\143

中级

实用指数：★★★★

实例说明

在一个数据库中可以创建多个存储过程，但与表相同，其中不能有相同的存储过程名。在 Java 程序中可以通过 SQL 语句来获取数据库中的所有存储过程。本实例实现查询 db_database06 下的所有存储过程名，将其在页面中显示，如图 6.4 所示。

显示数据库中的所有存储过程

编号	存储过程名
1	validateSelect
2	insertUser
3	selectUser

图 6.4　获取数据库中所有存储过程

可利用 SQL 语句查看数据库中的存储过程，语法如下：

```
select name from sysobjects where xtype   = 'p' and Status>0
```

参数说明

❶ sysobjects：该表为系统表，用于存储系统信息。

❷ xtype：sysobjects 系统表中的字段，指定了当前记录是何种类型的对象名，类型为 p 表示是存储过程。

❸ Status：sysobjects 系统表的状态标识字段，当该字段中的数值大于 0 时表示当前记录所表示的对象为用户所有，否则为系统所有。

注意： 获取数据库中所有存储过程是一条 SQL 语句，执行 SQL 语句需要使用 Statement 对象，而不是调用存储过程的 CallableStatement 对象。

设计过程

（1）在项目中创建类 GainProcedure，在该类中定义操作数据库的相关方法。其中，定义获取数据库中所有存储过程方法 executeGain()，该方法以 List 集合返回查询结果集。具体代码如下：

```
public List executeGain() {
    conn = getConn();                                               //获取数据库连接
    Statement cs = null;
    String sql = "select name from sysobjects where xtype = 'p' and status > 0";   //定义调用存储过程的 SQL 语句
    List list = new ArrayList();                                    //定义用于返回值的集合对象
    try {
        cs = conn.createStatement();
        ResultSet rest = cs.executeQuery(sql);                      //执行 SQL 语句
        while (rest.next()) {                                       //循环遍历查询结果集
            String name = rest.getString(1);                        //获取查询结果集中数据
            list.add(name);                                         //将数据添加到集合中
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return list;                                                    //返回查询结果集
}
```

（2）在项目中创建名为 GarinServlet 的 Servlet，在该 Servlet 中调用获取数据库中所有存储过程的方法，并将查询结果保存在 request 范围内。具体代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    GainProcedure procedure = new GainProcedure();                  //创建保存操作数据库方法的类对象
    List list = procedure.executeGain();                            //调用获取所有数据库中存储过程的方法
    request.setAttribute("list",list);                              //将查询结果保存到 request 对象中
    request.getRequestDispatcher("list.jsp").forward(request, response);   //设置请求转发地址
    }
```

（3）在 list.jsp 页面中以表格的形式显示查询结果，具体代码如下：

```
<table width="303" height="71" border="1">
  <tr>
    <td width="118" height="37"><div align="center">编号</div></td>
    <td width="169"><div align="center">存储过程名</div></td>
  </tr>
  <%
    List list = (List)request.getAttribute("list");               //获取保存在 request 对象中的信息
    for(int i = 0;i<list.size();i++){                              //循环遍历集合对象
  %>
 <tr>
   <td height="35"><div align="center"><%=i+1 %></div></td>
   <td><div align="center"><%=list.get(i)%></div></td>
 </tr>
 <%} %>
</table>
```

心法领悟 143：解析存储过程的命名标准。

与 Java 程序相同，存储过程也有自己的命名标准。虽然不一定所有程序员都按照统一的标准，但是养成好

的编程习惯是很有必要的。存储过程的命名标准为：

- 所有的存储过程均要以 proc 为前缀，系统存储过程以“sp_”为前缀。
- 表名就是存储过程访问的对象。
- 最后的行为动词就是存储过程要执行的任务。

例如，存储过程 procUserSelect。

实例 144　修改存储过程　高级

光盘位置：光盘\MR\06\144　实用指数：★★★★

实例说明

通常情况下，修改已经创建的存储过程需要打开 SQL Server 企业管理器，而本实例却实现了在 JSP 中修改已经创建的存储过程。运行本实例，在如图 6.5 所示的文本框中输入修改存储过程的 SQL 语句，单击“修改存储过程”按钮，即可修改指定的存储过程。

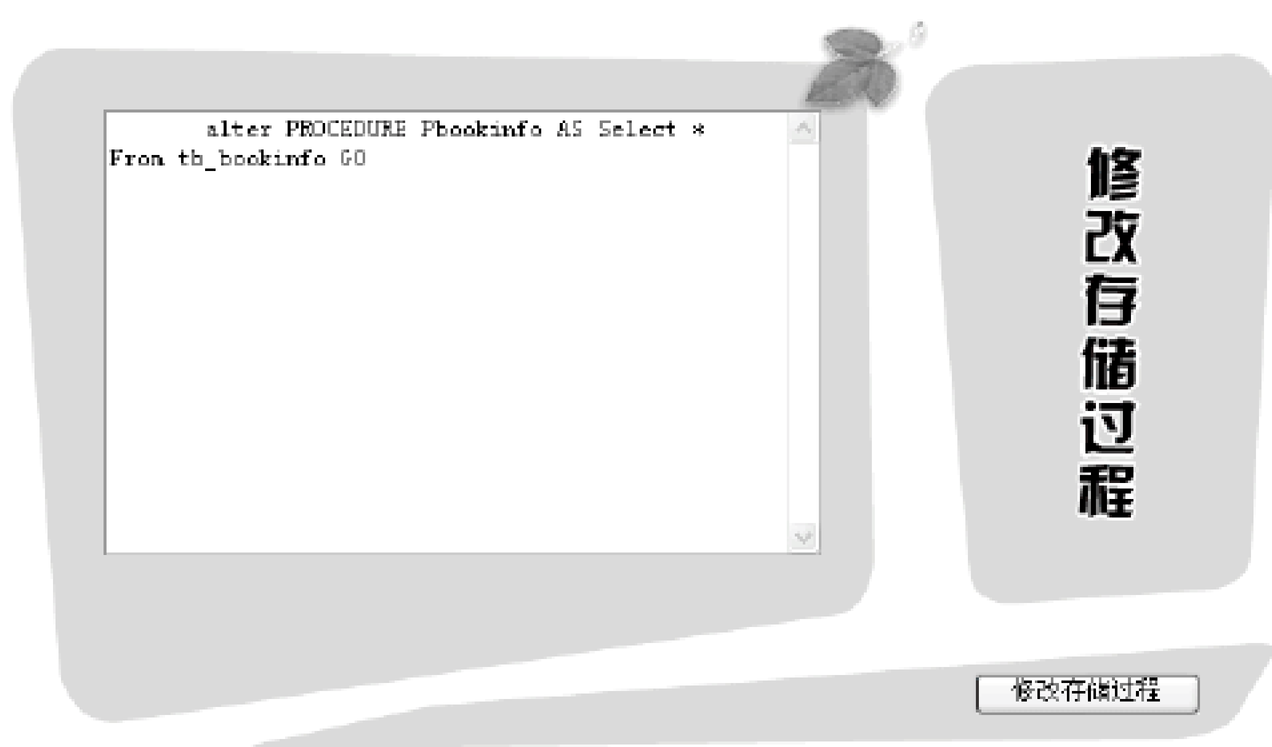


图 6.5　修改存储过程

关键技术

本例主要利用 SQL 语句中的 ALTER PROCEDURE 语句来实现。

语法如下：

```
ALTER PROCEDURE [ EDURE ] procedure_name [ ; number ]
    [ { @parameter data_type }
      [ VARYING ] [ = default ] [ OUTPUT ]
    ] [ ,...n ]
[ WITH
    { RECOMPILE | ENCRYPTION
        | RECOMPILE , ENCRYPTION
    }
]
[ FOR REPLICATION ]
AS
    sql_statement [ ...n ]
```

参数说明

❶ procedure_name：要更改的存储过程的名称。

❷ number：现有的可选整数，该整数用来对同一名称的过程进行分组，这样可以用一条 DROP PROCEDURE 语句将对应过程全部删除。

❸ @parameter：过程中的参数。

❹ data_type：参数的数据类型。

❺ VARYING：指定作为输出参数支持的结果集。

❻ default：参数的默认值。

❼ OUTPUT：表明参数是返回参数。

❽ n：最多可指定 2～100 个参数的占位符。

❾ RECOMPILE：表明 Microsoft® SQL Server™ 不会高速缓存该过程的计划，该过程将在运行时重新编译。

❿ ENCRYPTION：表示 SQL Server 加密 syscomments 表中包含 ALTER PROCEDURE 语句的条目。使用 ENCRYPTION 可防止将过程作为 SQL Server 复制的一部分发布。

（1）创建 UserDao 类，定义连接、更新和关闭数据库的方法。

（2）创建 index.jsp 页面，通过 JavaBean 标签调用 UserDao 类，应用 executeUpdate()方法执行修改存储过程的操作。创建 form 表单，通过文本域提交修改存储过程的 SQL 语句。其关键代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java"
        import="java.sql.*" errorPage=""%>
<jsp:useBean id="dao" class="com.pkh.dao.UserDao" scope="request"></jsp:useBean>
<%
        if ("修改存储过程".equals((String)request.getParameter("Submit")) && request.getParameter("textarea")!=""){
                boolean i = dao.executeUpdate((String)request.getParameter("textarea"));
                if (i){
                        out.println("<script type='text/javascript'>window.alert('存储过程修改成功！ ');</script>");
                }else{
                        out.println("<script type='text/javascript'>window.alert('存储过程修改失败！ ');</script>");
                }
        }
%>
```

秘笈心法

心法领悟 144：显式设置列的 NULL 或 NOT NULL。

建议在存储过程的任何 CREATE TABLE 或 ALTER TABLE 语句中都为每列显式指定 NULL 或 NOT NULL，例如，在创建临时表时。可以通过 ANSI_DFLT_ON 和 ANSI_DFLT_OFF 选项控制 SQL Server 为列指派 NULL 或 NOT NULL 特性的方式（如果在 CREATE TABLE 或 ALTER TABLE 语句中没有指定）。如果某个连接执行的存储过程对这些选项的设置与创建该过程的连接的设置不同，则为第 2 个连接创建的表列可能会有不同的为空性，并且表现出不同的行为方式。如果为每个列显式声明了 NULL 或 NOT NULL，那么将对所有执行该存储过程的连接使用相同的空值创建临时表。

实例 145	删除存储过程 光盘位置：光盘\MR\06\145	中级 实用指数：★★★★

实例说明

对于数据库中的错误数据或者无用的存储过程，一般情况下要将其删除，以便节省数据库的资源。本实例主要实现在 JSP 中动态删除存储过程（使用 CREATE PROCEDURE 创建的）的功能。运行程序，在如图 6.6 所示页面中选择要删除的存储过程，单击“删除”按钮，如果删除成功，就会弹出“存储过程删除成功”的提示对话框。



图 6.6　删除存储过程

关键技术

本实例主要是用 SQL 语句中的 DROP PROCEDURE 语句实现的。DROP PROCEDURE 语句用来删除一个或多个存储过程或者过程组，其语法如下：

```
DROP PROCEDURE { procedure } [ ,...n ]
```

参数说明

❶ procedure：表示要删除的存储过程或存储过程组的名称。过程名称必须符合标识符规则。

❷ n：表示可以指定多个过程的占位符。

（1）创建 UserDao 类，定义连接、更新、查询和关闭数据库的方法。

（2）创建 index.jsp 页面，通过 JavaBean 标签调用 UserDao 类，应用 executeUpdate()方法执行删除存储过

程的操作，应用 selectStatic()方法查询出数据库中的所有存储过程。创建 form 表单，为每个存储过程添加一个单选按钮，用于实现删除操作。其关键代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java"
        import="java.sql.*" errorPage=""%>
<jsp:useBean id="dao" class="com.pkh.dao.UserDao" scope="request"></jsp:useBean>
<%
        if (request.getParameter("Submit")!=null && request.getParameter("radiobutton")!=null){
                boolean i = dao.executeUpdate("DROP PROCEDURE " + (String)request.getParameter("radiobutton"));
                if (i){
                        out.println("<script type='text/javascript'>window.alert('存储过程删除成功！');</script>");
                }else{
                        out.println("<script type='text/javascript'>window.alert('存储过程删除失败！');</script>");
                }

        }
%>
<%
        ResultSet Rs = dao.selectStatic("Select * From sysobjects where xtype='P'");          //执行查询操作
       while (Rs.next()){                                                                      //循环输出查询结果集
%>
        <tr>
                <td><div align="left">
                        <input type="radio" name="radiobutton" value="<%=Rs.getString("name")%>">
                        <%=Rs.getString("name")%></div></td>
        </tr>
<%
        }
%>
```

秘笈心法

心法领悟 145：在 SQL Server 中执行存储过程。

本章中介绍的是在 Java 程序中调用存储过程，存储过程作为数据库对象，可以在数据库下直接被调用。在数据库中执行存储过程使用 execute 关键字即可。例如，要执行存储过程 proUserSelect，可使用 execute proUserSelect 语句。

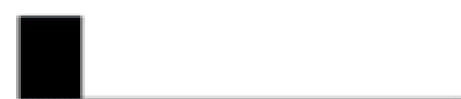
6.2　使用触发器

实例 146	应用触发器添加日志信息 光盘位置：光盘\MR\06\146	高级 实用指数：★★★★

实例说明

触发器是一种特殊类型的存储过程，是为响应数据库操作语句 DML 事件或数据定义语言 DDL 事件而执行的存储过程。当用户对表进行相应操作时，触发器启动执行。触发器可以基于表，也可以基于视图。SQL 支持 3 种触发器，即 INSERT、UPDATE 和 DELETE。本实例应用的是 INSERT 触发器，实现当向用户表中插入信息后，同时将该用户的登录时间添加到日志表（tb_info）中。本实例的运行结果如图 6.7 所示。

图 6.7　应用触发器添加日志信息



要实现本实例，首先需要在数据库中创建触发器，这样当执行相应的数据操作后，会自动调用触发器。可以使用 SQL 语句中的 CREATE

TRIGGER 语句来创建触发器，其语法如下：

```
CREATE TRIGGER trigger_name
ON { table | view }
[ WITH ENCRYPTION ]
{
    { { FOR | AFTER | INSTEAD OF } { [ INSERT ] [ , ] [ UPDATE ] }
        [ WITH APPEND ]
        [ NOT FOR REPLICATION ]
        AS
        [ { IF UPDATE ( column )
            [ { AND | OR } UPDATE ( column ) ]
                [ ...n]
        | IF ( COLUMNS_UPDATED ( ) { bitwise_operator } updated_bitmask )
                { comparison_operator } column_bitmask [ ...n]
        } ]
        sql_statement [ ...n]
    }
}
```

参数说明

❶ trigger_name：所要创建的触发器的名称。

❷ table|view：创建触发器所在的表或视图，也可以称为触发器表或触发器视图。

❸ AFTER：设定触发器只有在完成指定的所有 SQL 语句之后才会被触发。

❹ AS：触发器要执行的操作。

❺ sql_statement：触发器的条件或操作。触发器条件指定其他准则，以确定 DELETE、INSERT 或 UPDATE 语句是否导致执行触发器。

本实例实现在 SQL Server 数据库下创建触发器，该数据库中有两个非常有用的临时表，分别为 INSERTED 和 DELETED。当由插入操作产生触发器时，会将插入的数据先存储在 INSERTED 临时表中；当由删除操作产生触发器时，会将删除的数据先存储在 DELETED 临时表中；当由更新操作产生触发器时，会将更新前的数据存储在 DELETED 临时表中，而将更新后的数据存储在 INSERTED 临时表中。了解了这两个临时表，对掌握本章中涉及的触发器知识非常重要。

（1）首先在数据库中创建触发器，当用户在用户表中添加数据时，系统会自动执行触发器，向日志表中添加信息。创建触发器的代码如下：

```
create trigger triInfoInsert on tb_user
for insert
as
 declare @leavePerson varchar(20)
 select @leavePerson = userName from inserted
 insert tb_info (name,ddate) values (@leavePerson,getDate())
```

（2）在项目中创建类 UserUtil，在该类中定义 executeUpdate()方法，实现向 tb_user 表中添加数据。当系统调用该方法时，会自动触发触发器。该方法具体代码如下：

```
public boolean executeUpdate(User user) {
        conn = getConn();                                                    //获取数据库连接
        try {
                PreparedStatement statement = conn
                                .prepareStatement("insert into tb_user values(?,?,?,?,?)");    //定义添加数据的 SQL 语句
                statement.setString(1, user.getUserName());                  //设置预处理语句的参数值
                statement.setString(2, user.getPassWord());
                statement.setInt(3, user.getAge());
                statement.setString(4, user.getSex());
                statement.setString(5, user.getJob());
                statement.executeUpdate();                                   //执行预处理语句
                return true;
        } catch (Exception e) {
                e.printStackTrace();
                return false;
        }
}
```

秘笈心法

心法领悟 146：触发器限制。

触发器虽然应用起来很简单，但是也有一定的限制。首先 CREATE TRIGGER 必须是批处理中的第 1 条语句，并且只能应用到一张表中。触发器只能在当前的数据库中创建，不过可以引用当前数据库的外部对象。如果一张表的外键在 DELETE 或 UPDATE 操作上定义了级联，则不能在该表上定义 INSTEAD OF DELETE 或 UPDATE 触发器。



实例 147　应用触发器级联删除数据

光盘位置：光盘\MR\06\147

高级

实用指数：★★★★

实例说明

在实际开发中，经常会涉及两张有关联的表，例如，学生表与学生成绩表。当对一张表进行操作时，就需要考虑到另一张表的数据变化。对于这种情况，应用触发器是很方便的，也可以很好地维护数据完整性，避免由于程序员的疏忽而导致错误的出现。本实例实现创建 DELETE 触发器，当用户在学生表中删除数据时，学生成绩表中对应的数据也会被删除。本实例运行结果如图 6.8 所示。



图 6.8　应用触发器级联删除数据

本实例实现的是创建 DELETE 触发器，其工作流程如下。

当触发 DELETE 触发器后，从特定表中删除的行将被放置到一个特殊的 DELETED 表中。DELETED 表是一个逻辑表，用于保留已被删除数据行的一个副本。DELETED 表还允许引用由初始化 DELETE 语句产生的日志数据。

使用 DELETE 触发器时，需要考虑以下的事项和原则。

- 当某行被添加到 DELETED 表中时，就不再存在于数据表中，因此，DELETED 表和数据库没有相同的行。
- 创建 DELETED 表时，控件是从内存中分配的。DELETED 表总是被存储在高速缓存中。

注意：在 SQL Server 中使用 TRUNCATE TABLE 语句删除表中所有行时，不会触发 DELETE 触发器。

设计过程

（1）在数据库中定义 DELETE 触发器，实现当在学生表中删除记录时，将触发该触发器，将学生成绩表中的相应数据删除。具体代码如下：

```
create trigger triGradeDelete on tb_stu
for delete
 as
  declare @id int
  select @id =    id from deleted
  delete from tb_grade where tb_grade.sid = @id
```

（2）在项目中定义 DeteleUtil 类，在该类中定义从学生表中删除数据的方法 deleteGrade()。具体代码如下：

```
public void deleteGrade(int id){
        conn = getConn();                                          //调用获取数据库连接方法
        try {
                Statement statement = conn.createStatement();      //创建 Statement 对象
                String sql = "delete from tb_stu where id = "+id;  //定义删除数据的 SQL 语句
                statement.executeUpdate(sql);                      //执行删除语句
        } catch (Exception e) {
                e.printStackTrace();
        }
}
```

秘笈心法

心法领悟 147：慎用触发器。

触发器功能强大，可轻松、可靠地实现很多复杂的功能，为什么又要慎用呢？触发器本身没有问题，但如果滥用就会造成数据库及应用程序的维护困难。在数据库操作中，可以通过关系、触发器、存储过程、应用程序来实现数据操作；同时规则、约束、默认值也是保证数据完整性的重要保障。如果过分依赖触发器，势必影响数据库的结构，同时也增加了维护的难度。

实例 148　调用 UPDATE 触发器修改数据　高级

光盘位置：光盘\MR\06\148　实用指数：★★★★

实例说明

在特定的表上执行 UPDATE 语句时，将触发 UPDATE 触发器。UPDATE 操作包括两个部分：将需要更新的内容从表中删除，然后插入新值。因此，UPDATE 触发器同时涉及删除表中数据和插入表中数据两项内容。本实例调用的就是 UPDATE 触发器，当用户修改教师表中数据时，选课表中数据也会随之修改。运行本实例，首先会将教师表中所有信息在页面中显示，如图 6.9 所示。用户可通过单击页面中的“修改”超链接修改教师的信息，如图 6.10 所示。

教师表中数据			
编号	姓名	学科	修改
2	王宁	信息技术	
3	小婉	计算机	
4	陈双	语文	
5	春雷	英语	

图 6.9　教师表中数据

编 号：	3
姓 名：	小婉
课 程：	计算机
	确定修改

图 6.10　修改页面

关键技术

本实例的关键是在数据库中成功地创建 UPDATE 触发器，这样当通过 Java 程序修改数据时，将自动执行触发器。

UPDATE 触发器的工作过程：可将 UPDATE 语句看成两步操作，即捕获数据前的 DELETE 语句和捕获数

据后的 INSERT 语句。当在定义有触发器的表上执行 UPDATE 语句时，原始行将被移入到 DELETED 表，更新行被移入到 INSERTED 表。UPDATE 触发器通过检查 DELETED 表、INSERTED 表以及被更新的表，来确定是否更新了多行以及如何执行触发器动作。

设计过程

（1）在数据库中创建 UPDATE 触发器，实现当在 tb_teacher 表中修改数据时，选课表 tb_elective 中对应的数据也随之更改。创建 UPDATE 触发器的代码如下：

```
create trigger triteacher on tb_teacher
for update
as
 declare @name varchar(10),@course varchar(20),@tName varchar(20),@newName varchar(20),@id int
 begin
    select @course = course from DELETED
    select @name = course from INSERTED
    select @tName = tName    from DELETED
    select @newName = tName from INSERTED
    select @id = id from INSERTED
    update tb_elective set elective = @name,teacher= @newName where id = @id
 end
```

（2）在项目中创建 TeacherUtil 类，在该类中定义操作数据库的各种方法，然后定义按照指定编号查询教师表中数据的方法 getTeacherById()，该方法以 int 对象为参数。具体代码如下：

```
public Teacher getTeacherById(int id){
        conn = getConn();                                    //获取数据库连接方法
        String sql = "select * from tb_teacher where id= "+id;    //定义查询 SQL 语句
        Teacher teacher = new Teacher();                     //创建与数据表对应的 JavaBean 对象
        try {
                Statement statement = conn.createStatement();     //创建 Statement 对象
                ResultSet rest = statement.executeQuery(sql);     //执行查询语句，获取查询结果集
                while(rest.next()){                              //循环遍历查询结果集
                        teacher.setId(rest.getInt(1));           //应用查询结果集中数据设置对象属性
                        teacher.settName(rest.getString(2));
                        teacher.setCourse(rest.getString(3));
                }
        } catch (SQLException e) {
                e.printStackTrace();
        }
        return teacher;                                      //返回查询结果
 }
```

（3）在 TeacherUtil 类中定义 updateTeacher()方法（该类包含有 Teacher 类对象的参数，表示要进行修改的教师），当该方法被执行时，将触发触发器。具体代码如下：

```
public void updateTeacher(Teacher teacher){
      conn = getConn();                                                                    //获取数据库连接
      try {
          PreparedStatement statement = conn.prepareStatement("update tb_teacher set tName=?,course = ? where id = ?") ;//定义 PreparedStatement 对象
          statement.setString(1, teacher.getName());                                       //设置预处理语句的参数
          statement.setString(2, teacher.getCourse());
          statement.setInt(3, teacher.getId());
          statement.executeUpdate();                                                       //执行修改操作
      } catch (SQLException e) {
          e.printStackTrace();
      }
}
```

心法领悟 148：在数据库中定义变量。

本实例使用的是 SQL Server 数据库，在创建触发器时，定义了变量@name、@course、@tName、@newName 和@id，它们都属于局部变量。在 SQL Server 中，命名变量必须以“@”开头，并且定义局部变量还需要使用 DECLARE 语句。语法如下：

```
DECLARE
{
```

```
    @varaible_name datatype[,…n]
}
```

其中，@varaible_name 表示变量的名称；datatype 表示变量的数据类型。

实例 149 获取数据库中所有触发器名称

光盘位置：光盘\MR\06\149

高级

实用指数：★★★

实例说明

本实例实现的是在 JSP 中获取 SQL Server 数据库中所有的触发器，运行结果如图 6.11 所示。

获取数据库中的触发器
reminder
messagelog

图 6.11　获取数据库中所有触发器

关键技术

本实例使用 SQL 语句查看数据库中所有的触发器。语法如下：

```
Select xtype From sysobjects Where xtype = 'TR'
```

参数说明

❶ sysobjects：该表为系统表，用于存储系统信息。

❷ xtype：sysobjects 系统表中的字段，指定了当前记录是何种类型的对象名，类型为 TR 表示是触发器。

设计过程

（1）创建 UserDao 类，定义连接、查询和关闭数据库的方法。

（2）创建 index.jsp 页面，通过 JavaBean 标签调用 UserDao 类，应用 selectStatic()方法执行查询数据库中触发器的操作，并通过 while 语句循环输出查询结果集。其关键代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java"
        import="java.sql.*" errorPage=""%>
<jsp:useBean id="dao" class="com.pkh.dao.UserDao" scope="request"></jsp:useBean>
<%
        ResultSet Rs = dao.selectStatic("Select * From sysobjects Where xtype = 'TR'");
        while (Rs.next()){
%>
<tr>
        <td height="20" bgcolor="#66CCFF"><div align="center"><%=Rs.getString("name")%></div></td>
</tr>
<% } %>
```

秘笈心法

心法领悟 149：begin…end 语句块。

begin…end 语句用于将多个 Transact-SQL 语句组合为一个逻辑块。当流程控制语句必须执行一个包含两条或两条以上的 Transact-SQL 语句块时，可以使用 begin…end 语句。begin…end 语句必须成对使用，任何一条语句均不能单独使用。begin 语句后为 Transact-SQL 语句块，最后是 end 语句，用于指示语句块结束。

实例 150 创建带有触发条件的触发器

光盘位置：光盘\MR\06\150

中级

实用指数：★★★★

与查询语句一样，在触发器中也可以添加触发条件，从而确保只有在满足一定的条件时才会执行相应的操作。本实例实现的创建带有触发条件的触发，涉及两张表，分别为学生表与学生成绩表。当向学生成绩表中添加数据时，会触发触发器；只有当要添加的信息在学生信息表中存在时，才允许添加。本实例的运行结

果如图 6.12 所示。



图 6.12　创建带有触发条件的触发器

关键技术

当通过如图 6.12 所示窗体向数据库中添加数据时，将触发 triStuSelect 触发器。如果添加的学生姓名在学生信息表中不存在，将执行 rollback transaction 语句，取消操作。可以基于一张表创建多个触发器，DBMS 把同一个表中所有触发器看作同一事务的一部分。因此，只要其中一个触发器执行了 ROLLBACK TRANSACTION 语句，那么所有的操作都将被取消。

注意：一个触发器只能作用在同一张表上。

在存储过程中添加 if 语句的语法为：

```
If<条件表达式>
{命令行|程序块}
```

其中，“条件表达式”可以是各种表达式的组合，但表达式的值必须是逻辑值“真”或“假”；“命令行”和“程序块”可以是合法的 Transact-SQL 任意语句，但含两条或两条以上语句的程序块必须加 begin…end 子句。

（1）在数据库中创建带有触发条件的触发器 triStuSelect，实现当在学生成绩表中添加数据时，会在学生信息表中进行查询，如果存在对应的信息，允许插入，否则不允许插入。具体代码如下：

```
create trigger triStuSelect
 on tb_grade
 after insert as
  declare @sid int
  select @sid = sid from inserted
  if(@sid not in (select id from tb_stu))
begin
 rollback transaction
 print ('输入的学生标号错误，请重新输入')
end
```

（2）在项目中创建类 GradeUtil，在该类中定义操作数据的方法，然后定义向学生成绩表中添加数据的方法 executeUpdate()，该方法以 int 形式返回插入数据的行数。具体代码如下：

```
public int executeUpdate(Grade grade) {
        conn = getConn();                                                                    //获取数据库连接
        int count = 0;
        try {
                PreparedStatement statement = conn
                                .prepareStatement("insert into tb_grade(sid,english,chinese,math) values(?,?,?,?)");   //定义添加数据的 SQL 语句
```

```
            statement.setInt(1, grade.getSid());                          //设置预处理语句参数值
            statement.setFloat(2, grade.getEnglish());
            statement.setFloat(3, grade.getChinese());
            statement.setFloat(4, grade.getMath());
        count =  statement.executeUpdate();                              //执行预处理语句
            return count;                                                //返回插入数据的函数
    } catch (Exception e) {
            e.printStackTrace();
            return count;
    }
}
```

（3）当用户在 index.jsp 页面中完成信息填写后，单击“添加”按钮，系统会在 GradeServlet 中处理请求，实现添加操作，具体代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    GradeUtil util = new GradeUtil();                                    //创建保存有操作数据的类对象
    Grade grade = new Grade();                                           //创建与数据表对应的 JavaBean 对象
    grade.setSid(Integer.parseInt(request.getParameter("sidTextfield")));    //设置对象属性
    grade.setChinese(Float.parseFloat(request.getParameter("chineseTextfield")));
    grade.setEnglish(Float.parseFloat(request.getParameter("englishTextfield")));
    grade.setMath(Float.parseFloat(request.getParameter("mathTextfield")));
    int   count = util.executeUpdate(grade);                             //调用插入数据方法
    String message = "数据添加失败！！ ";
    if(count > 0){                                                       //如果数据插入成功
            message = "数据添加成功！！ ";
    }
    request.setAttribute("message",message);                             //将信息保存在 request 对象中
    request.getRequestDispatcher("index.jsp").forward(request, response);    //设置请求转发地址
}
```

秘笈心法

心法领悟 150：创建临时表。

临时表常用来保存中间结果，其名称前带有“#”。临时表只存在于存储过程被创建时获取用户会话期间。创建临时表可以使用 CREATE TABLE 语句，语法如下：

```
CREATE TABLE #temp(int x,inty)
```

其中，x、y 分别表示临时表的字段。

6.3 使用批处理

实例 151 使用批处理删除数据

光盘位置：光盘\MR\06\151

中级

实用指数：★★★★

如今的数据库处理数据时的速度快得惊人，单次执行的吞吐量非常大，执行效率非常高。这时速度的瓶颈主要出现在数据库的连接传输上。在 Java 中，每当需要执行 SQL 语句时都要创建一个数据库连接，并把要执行的 SQL 语句传送到数据库服务器，时间都消耗在了数据库的连接传输上。如果把要执行的 SQL 语句装载到一起，一次性发送给数据库执行，则会大大提高执行效率。本实例以操作收件箱为例，讲解了如何使用 JDBC 批量处理数据。

运行本实例，首先进入收件箱，如图 6.13 所示。选中要进行操作的邮件后单击“移至垃圾箱”按钮，选中的邮件将被批量移动到垃圾邮件箱；单击“彻底删除”按钮，所选邮件将被批量删除。



图 6.13　使用批处理删除数据

本实例中分别使用了 Statement 接口和 PreparedStatement 接口中的批量处理数据方法对数据库进行操作，下面就其关键技术进行讲解。

（1）addBatch()方法

addBatch()是 Statement 接口中的方法，用于将给定的 SQL 语句添加到 Statement 对象当前命令列表中。声明语法如下：

```
void addBatch(String sql) throws SQLException
```

参数说明

sql：INSERT 或 UPDATE 语句。

向 Statement 对象添加 SQL 语句的方法如下：

```
Statement st = …                                                          //省略部分代码
st.addBatch("INSERT INTO tableName(column1,column2) value('aa','bb')");  //添加 SQL 语句
st.addBatch("INSERT INTO tableName(column1,column2) value('cc','dd')");  //添加 SQL 语句
```

另外，PreparedStatement 接口中声明了一个没有参数的 addBatch()方法，其声明语法如下：

```
void addBatch() throws SQLException
```

向 PreparedStatement 对象添加 SQL 语句的方法如下：

```
Connection con =…                                                         //省略创建数据库连接代码
PreparedStatement ps = con.prepareStatement("DELETE FROM tb_inbox WHERE id = ?");
ps.setInt(1);                                                             //为“?”参数赋值
ps.addBatch();                                                            //添加到列表
ps.setInt(2);                                                             //为“?”参数赋值
ps.addBatch();                                                            //添加到列表
```

（2）executeBatch()方法

executeBatch()是 Statement 接口中定义的方法，用于将一批命令提交给数据库来执行。PreparedStatement 接口从 Statement 接口继承 executeBatch()方法。该方法的声明语法如下：

```
int[] executeBatch() throws SQLException
```

在本实例中将命令提交给数据库执行的关键代码如下：

```
Statement st = …                                                          //省略部分代码
st.addBatch("INSERT INTO tableName(column1,column2) value('aa','bb')");  //添加 SQL 语句
st.addBatch("INSERT INTO tableName(column1,column2) value('cc','dd')");  //添加 SQL 语句
st.executeBatch();                                                        //执行全部 SQL 语句
```

注意： 要想批量执行 SQL 语句，必须调用 Connection 对象的 setAutoCommit(false)方法将数据事务设置为手动提交模式。

另外，在本实例使用了 request.getParameterValues("name")方法得到一批用户选中的复选框的值。关键代码

如下：

```
String[] str = null;                                          //声明 String 数组
str = request.getParameterValues("checkbox");                 //获得选中复选框的值
```

(1) 新建 DBConnection 类，用于创建数据库连接；新建 InboxDto 类，用于封装数据记录。这两个类不属于本节重点内容，在此不再赘述，实现方法参见配书光盘中的源代码。

(2) 新建 InboxDao 类，用于操作数据库中收件箱对应的表。在 InboxDao 类中编写 findAll()方法，用于查询所有邮件。关键代码如下：

```
ResultSet rs = st.executeQuery("SELECT * FROM tb_inbox ORDER BY date desc");      //按发件时间降序排列
while(rs.next()){
        InboxDto mi = new InboxDto();                                              //声明 InboxDto 对象
        mi.setId(rs.getInt("id"));                                                 //为编号赋值
        mi.setTitle(rs.getString("title"));                                        //为标题赋值
        mi.setContent(rs.getString("content"));                                    //为内容赋值
        mi.setDate(rs.getTimestamp("date").toString());                            //为发件时间赋值
        mi.setAddresser(rs.getString("addresser"));                                //为发件人地址赋值
        list.add(mi);                                                              //加入到 list 列表
}
……//省略部分代码
```

(3) 编写 moveToCgbox()方法，将要移动的记录插入到目标表中，并将原表中的记录删除。关键代码如下：

```
//将邮件从收件箱移动到垃圾箱
public void moveToCgbox(String[] id){
        con = DBConnection.getConnection();                                        //获取数据库连接
        PreparedStatement ps = null;                                               //声明 PreparedStatement 对象
        Statement st = null;                                                       //声明 Statement 对象
        ResultSet rs = null;                                                       //声明 Statement
        try {
                con.setAutoCommit(false);                                          //设为手动提交方式
                st = con.createStatement();                                        //创建 Statement 对象
                ps = con.prepareStatement("SELECT * FROM tb_inbox WHERE id = ?");  //创建 ps 对象
                for (int i = 0; i < id.length; i++) {
                        ps.setInt(1, Integer.valueOf(id[i]));                      //按照编号查询记录
                        rs = ps.executeQuery();                                    //执行
                        rs.next();                                                 //将游标指向第一条记录
                        //生成 INSERT 语句，加入到 st 对象中
                        st.addBatch("INSERT INTO tb_cgbox (title,content,date,addresser) values " +
                                "('"+rs.getString("title")+"','"+rs.getString("content")+"','"+
                                rs.getTimestamp("date")+"','"+rs.getString("addresser")+"')");   //在垃圾箱中插入记录
                        st.addBatch("DELETE FROM tb_inbox WHERE id = " + id[i]);   //删除收件箱中的记录
        }
        st.executeBatch();                                                         //批量执行 st 对象中保存的 SQL 语句
        con.commit();                                                              //提交事务
        st.close();                                                                //关闭 st 对象
        rs.close();                                                                //关闭 rs 对象
        ps.close();                                                                //关闭 ps 对象
        } catch (Exception e) {
                try {
                        con.rollback();                                            //如果出现异常回滚
                }
        ……//省略部分代码
}
```

(4) 编写 delete()方法，用于批量删除收件箱中的邮件。关键代码如下：

```
public3 void delete(String[] id){
      PreparedStatement ps = null;
      try {
            con = DBConnection.getConnection();                                    //将事务提交方式设为手动提交
            con.setAutoCommit(false);                                              //关闭自动事务
            ps = con.prepareStatement("DELETE FROM tb_inbox WHERE id = ?");        //创建 ps 对象
            for (int i = 0; i < id.length; i++) {
                  ps.setInt(1, Integer.valueOf(id[i]));                            //为 ID 参数赋值
```

```
            ps.addBatch();                        //将参数添加到 PreparedStatement 对象的批处理命令中
        }
        ps.executeBatch();                        //将 ps 对象中的全部命令一起提交给数据库来执行
        con.commit();                             //手动提交事务
        ps.close();                               //关闭 ps 对象
    } catch (Exception e) {
        try {
            con.rollback();                       //如果发生异常，将事务回滚
        }
……//省略部分代码
```

（5）编写 showinbox.jsp 页面，显示收件箱中的邮件并在每个邮件前添加一个复选框。关键代码如下：

```
<html>
    <body>
        <h3 align="center">收件箱</h3>
        <form name="f1" action="" method="post" onsubmit="return issubmit()">
        <table align="center" width="700" border="1">
            <tr >
                <td>全选<input type="checkbox" name="all" onclick="selectAll()" /></td>
                <td>邮件主题</td>
                <td>发件人</td>
                <td>接收日期</td>
            </tr>
        <%
        InboxDto midto = null;
        List list = new InboxDao().findAll();
        for (int i = 0; i < list.size(); i++) {
            midto = (InboxDto) list.get(i);
        %>
            <tr>
                <td><input type="checkbox" name="checkbox" value='<%= midto.getId()%>' /></td>
                <td><%= midto.getTitle()%></td>
                <td><%= midto.getAddresser()%></td>
                <td><%= midto.getDate()%></td>
            </tr>
        <% } %>
            <tr>
                <td colspan="4" align="center">
                <input type="submit" value="移至垃圾箱" onclick="submitmove()" />
                <input type="submit" value="彻底删除"onclick="submitdelete()" />
                </td>
            </tr>
        </table>
        </form>
    </body>
</html>
```

秘笈心法

心法领悟 151：在 SQL Server 中输出文本。

在 Eclipse 中输出信息可以使用 System.out.print()语句；如果想在 SQL Server 的查询分析器中输出信息，可以使用 print 命令。print 命令可以输出指定的字符串、数字，也可以输出一个函数的返回值，或者一个字符串的表达式。例如，应用 print 关键字输出当前系统时间，代码如下：

```
print '当前系统时间为：'+ rtrim(convert (varchar(30),getdate()))
```



实例 152	批量提高员工工资 光盘位置：光盘\MR\06\152	高级 实用指数：★★★★

实例说明

由于在一个批处理语句中可以执行多条 SQL 语句，因此在实际开发中批处理得到了广泛应用。本实例应用

批处理技术来实现提升某些员工的工资，运行结果如图 6.14 所示。

选择	编号	姓名	部门	工资
☐	1	张静	Java开发部	5800
☐	2	李丽	VB开发部	5800
☑	3	刘梅	质量部	3300
☑	4	陈双	客户部	4760
☐	5	马丹	市场部	4500
	工资批量提升200元		工资批量提升400元	

图 6.14　批量提高员工工资

关键技术

本实例实现批处理技术使用的是 Statement 对象的 addBatch()方法与 executeBatch()方法，具体语法参见实例 151。

设计过程

在项目中创建类 EmpUtil，在该类中定义操作数据库的各种方法，然后定义批量更新数据的方法 updateBatch()。该方法有一个 String 类型的数组参数与一个 int 类型的参数，分别用于指定要修改工资的员工编号与提升的工资金额。具体代码如下：

```
public void updateBatch(String[] id, int laborage) {
    con = getConnection();                                              //获取数据库连接
    Statement cs = null;                                                //定义 Statement 对象
    try {
        cs = con.createStatement();                                     //实例化 Statement 对象
        for (int i = 0; i < id.length; i++) {
            cs.addBatch("update tb_emp set laborage = laborage +"
                    + laborage + " where id = '" + Integer.parseInt(id[i]) + "'");   //修改数据
        }
        cs.executeBatch();                                              //批量执行 SQL 语句
        cs.close();                                                     //将 Statement 对象关闭
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 152：SQL Server 数据库中的文本与图像类型。

在开发中根据实际需要定义数据类型是非常重要的，下面介绍一些特殊的数据类型。

- text：用于存储大量的文本数据，其容量理论上为 $1\sim2^{31}-1$（2147483647）字节，在实际应用时需要视硬盘的存储空间而定。
- ntext：与 text 类似，不同的是 ntext 采用 UNICODE 标准字符集，因此理论容量为 $2^{30}-1$（1073741823）字节。
- image：可变长度的二进制数据类型，最大长度为 $2^{31}-1$ 个字符。通常用来存储图像、声音等 OLE 对象。

实例 153　将教师表中数据全部添加到选课表

光盘位置：光盘\MR\06\153

中级

实用指数：★★☆

使用批处理可快速地完成相关的操作。例如，选课表和教师表是两个有关联关系的数据表，想要快速地将教师表中的数据添加到选课表中，就可以使用批处理来实现。本实例将使用批处理技术，将教师表中的数据全部添加到选课表中。本实例执行前选课表中数据为空，执行完毕后选课表中数据如图 6.15 所示。

id	elective	teacher	classRoom
11	计算机科学	张丹	待定
12	古典文学	赵华	待定
13	现代汉语	陈双	待定
14	儿童文学	李静	待定
15	微格教学	赵梅	待定

图 6.15　本实例的运行结果

关键技术

要实现本实例，首先要将教师表中的数据检索出来，再通过批处理将教师表中的数据添加到选课表。仍然是使用 Statement 接口中的 addBatch()方法与 executeBatch()方法。

设计过程

（1）在项目中创建类 BatchInsert，在该类中首先定义连接数据库的方法 getConn()，该方法以 Connection 对象作为返回值。具体代码参见配书光盘中的源程序，这里不再赘述。

（2）在该类中定义 executeTeacher()方法，实现获取教师表中所有数据。该方法以 List 作为返回值。具体代码如下：

```
public List executeTeacher() {
      conn = getConn();                                          //获取数据库连接
      Statement cs = null;                                       //定义 CallableStatement 对象
      String sql = "select * from tb_teacher";                   //定义调用存储过程的 SQL 语句
      List list = new ArrayList();
      try {
            cs = conn.createStatement();                         //实例化 Statement 对象
            ResultSet rest = cs.executeQuery(sql);               //执行 SQL 语句
            while (rest.next()) {                                //循环遍历查询结果集
                  Teacher teacher = new Teacher();               //定义与数据表对应的 JavaBean 对象
                  teacher.setId(rest.getInt(1));                 //设置对象的参数值
                  teacher.settName(rest.getString(2));
                  teacher.setCourse(rest.getString(3));
                  list.add(teacher);                             //向集合中添加对象
            }
      } catch (SQLException e) {
            e.printStackTrace();
      }
      return list;
}
```

（3）在该类中定义 addBatch()方法，批量将教师表中数据添加到选课表。具体代码如下：

```
public void insertBatch() {
      conn = getConn();                                          //获取数据库连接
      Statement cs = null;                                       //定义 Statement 对象
      try {
            cs = conn.createStatement();                         //实例化 Statement 对象
            List list = executeTeacher();
            for (int i = 0; i < list.size(); i++) {
                  Teacher teacher = (Teacher) list.get(i);
                  cs.addBatch("insert into tb_elective values ('"
                              + teacher.getCourse() + "','" + teacher.gettName()
                              + "','待定')");                      //添加 SQL 语句
            }
            cs.executeBatch();                                   //批量执行 SQL 语句
            cs.close();                                          //将 Statement 对象关闭
            conn.close();
      } catch (Exception e) {
            e.printStackTrace();
      }
}
```

心法领悟 153：利用 referer 请求头阻止“盗链”。

有一些站点自身没有提供下载空间，但是为了吸引人气、提高站点的访问量，也会提供一些软件的下载页面，并让下载的超链接指向其他站点上的资源。而真正提供下载的站点为了防止这种“盗链”，需要检查请求的来源，只接受本站内的页面链接，阻止其他站点的页面链接的下载请求。要实现这一功能，就需要检查请求消息的 referer 头字段是否与本站匹配。

实例 154　在批处理中使用事务

光盘位置：光盘\MR\06\154

高级

实用指数：★★★★

实例说明

在企业级的应用程序中，经常会遇到对多个数据表同时存取的情况。最明显的例子就是银行的转账业务，从汇款账户中减去指定金额，并将该金额添加至收款账户中。但如果在转账的过程中发生程序错误或者系统断电等意外情况，就可能导致汇款账户的余额已经减少而收款账户的余额还没有增加。这时就需要应用事务对该问题进行处理。本实例就是模拟银行转账系统，在如图 6.16 所示页面中选择转账的账户与转入的账户，然后输入转账的金额，单击“转账”按钮，即可完成转账操作。



图 6.16　在批处理中使用事务来处理转账问题

关键技术

在数据库系统中，实际上每一条 SQL 语句都是一个事务。当这条语句执行时，要么执行成功，要么执行失败，退回最初的状态。不过，如果执行一组 SQL 语句的操作，当其中某些步骤出现错误时，就不能还原到最初的状态了，这时就需要用到数据库的事务处理机制。

在 JDBC 中事务处理的一般步骤如下：

（1）调用 setAutoCommit()方法设置自动提交方式为 false。

语法：

```
setAutoCommit(false)
```

功能：更改 SQL 语句的提交方式。

（2）在异常处理中完成数据回滚。

语法：

```
rollback()
```

功能：将 SQL 操作回滚。

（3）手动提交 SQL 语句。

语法：

```
conn.commit()
```

功能：将成批的 SQL 操作提交给数据库。

（4）调用 setAutoCommit()方法恢复原来的提交方式。

（1）在项目中创建类 BatchAffair，在该类中定义操作数据库的各种方法，然后定义获取账户表 tb_transition 中所有账户的方法 selectIds()，该方法以 List 集合对象作为返回值。具体代码如下：

```
public List selectIds() {
    conn = getConn();                                        //获取数据库连接
```

```
    Statement cs = null;                                          //定义 CallableStatement 对象
    String sql = "Select accoutNumber from tb_transition";        //定义查询视图的 SQL 语句
    List list = new ArrayList();                                  //定义保存查询结果的 List 集合
    try {
        cs = conn.createStatement();                              //实例化 Statement 对象
        ResultSet rest = cs.executeQuery(sql);                    //执行 SQL 语句
        while (rest.next()) {                                     //循环遍历查询结果集
            String accoutNumber = rest.getString(1);
            list.add(accoutNumber);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return list;
}
```

（2）在 BatchAffair 类中定义转账方法 Batch()。该方法包含两个 String 类型的参数与一个 float 类型的参数，分别用于指定转账的账户与转入的账户以及转账的金额。具体代码如下：

```
public void Batch(String incomeId, String goId, float money) throws SQLException {
    try {
        conn = getConn();                                         //获取数据库连接
        boolean autoCommit = conn.getAutoCommit();
        conn.setAutoCommit(false);

        Statement cs = null;                                      //定义 Statement 对象
        cs = conn.createStatement();                              //实例化 Statement 对象
        cs.addBatch("update tb_transition set deposit = deposit-" + money
                + " ,transition = transition-" + money
                + " where accoutNumber = " + goId);               //定义修改转账表中数据的方法
        cs.addBatch("update tb_transition set deposit = deposit+" + money
                + " ,shift = shift+" + money + " where accoutNumber = "
                + incomeId);
        cs.executeBatch();                                        //批量执行 SQL 语句
        cs.close();                                               //将 Statement 对象关闭
        conn.commit();
        conn.setAutoCommit(autoCommit);
        conn.close();
    } catch (Exception e) {
        conn.rollback();
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 154：事务的 3 种运行模式。

- 自动提交事务：每条单独的语句都是一个事务。
- 显式事务：每个事务均以 begin transaction 语句显式开始，以 commit 或 rollback 语句显式结束。
- 隐性事务：在前一个事务完成时新事务隐式启动，但每个事务仍以 commit 或 rollback 语句显式完成。

6.4 使 用 视 图

实例 155	通过 Java Web 程序创建视图 光盘位置：光盘\MR\06\155	中级 实用指数：★★★★

实例说明

在网络程序的后台管理中，可能需要具有动态生成数据库视图的功能。本实例将演示如何动态创建数据库中的视图。运行程序，在文本框中输入要创建视图的 SQL 语句，单击“创建”按钮，即可创建视图并显示相应

的提示信息，如图 6.17 所示。



图 6.17　创建视图

关键技术

视图作为数据库对象并不存储数据，只是用来显示底层数据库的数据信息。可以将视图看作以后进行查询的来源。与任何表一样，视图可以通过在 SELECT 语句和 FROM 子句中引用一个视图来查询。视图可以引用表中的一列，也可以引用指定表中的任意多列。创建视图的语法如下：

```
CREATE VIEW view_name[(column[,…n])]
[WITH ENCRYPTION]
AS
select_statement
[WITH CHECK OPTION]
```

参数说明

❶ view_name：视图的名称。

❷ column：定义视图的字段名，如果没有指定，则视图字段将获得与 SELECT 语句中的字段相同的名称。

❸ WITH ENCRYPTION：指定将 CREATE VIEW 语句文本存储到系统表时进行加密，加密以后，任何人都不能通过系统存储过程或其他方法从系统表中检索视图定义文本。

❹ AS：视图要执行的操作。

❺ select_statement：定义视图的查询语句。该语句可以引用多个表或其他视图。

❻ WITH CHECK OPTION：规定在视图上执行的所有数据修改语句都必须符合由 select_statement 设置的准则。通过视图修改记录，WITH CHECK OPTION 可确保提交修改后，仍可通过视图看到修改的数据。

在 CREATE VIEW 语句中，对于查询语句有以下的限制。

- ❑ 创建视图的用户必须对该视图所参照或引用的表或视图具有适当的权限。
- ❑ 不能引用临时表。

在查询语句中，不能包含 ORDER BY、COMPUTE 或 COMPUTER BY 关键字，也不能包含 INTO 关键字。

（1）创建 UserDao 类，定义连接、更新和关闭数据库的方法。其关键代码如下：

```
public class UserDao {
        String url = "jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_database08";
        String username = "sa";
        String password = "";
        private Connection con = null;
        private Statement stmt = null;
        private ResultSet rs = null;
        public UserDao() {                                                    //通过构造方法加载数据库驱动
                try {
                        Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
                } catch (Exception ex) {
                        System.out.println("数据库加载失败");
                }
        }
    public boolean Connection() {                                             //创建数据库连接
        try {
            con = DriverManager.getConnection(url, username, password);
```

```
        } catch (SQLException e) {
            System.out.println(e.getMessage());
            System.out.println("creatConnectionError!");
        }
        return true;
    }
    public ResultSet selectStatic(String sql) throws SQLException {                    //对数据库的查询操作
            ResultSet rs=null;
            if (con == null) {
                Connection();
            }
            try {
                stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);
                rs = stmt.executeQuery(sql);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            closeConnection();
            return rs;
    }
    public boolean executeUpdate(String sql) {                                         //更新数据库
        if (con == null) {
            Connection();
        }
        try {
            stmt = con.createStatement();
            int iCount = stmt.executeUpdate(sql);
            System.out.println("操作成功，所影响的记录数为" + String.valueOf(iCount));
        } catch (SQLException e) {
            System.out.println(e.getMessage());
            return false;
        }
        closeConnection();
        return true;
    }
    public void closeConnection() {                                                    //关闭数据库的操作
       if (con != null && stmt != null && rs != null) {
            try {
                rs.close();
                stmt.close();
                con.close();
            } catch (SQLException e) {
                e.printStackTrace();
                System.out.println("Failed to close connection!");
            } finally {
                con = null;
            }
        }
    }
}
```

（2）创建 index.jsp 页面，通过 form 表单编写创建视图语句，调用 UserDao 类中的 executeUpdate()方法执行创建视图的操作。其关键代码如下：

```
<%
    if ("创建".equals(request.getParameter("Submit")) && request.getParameter("textarea")!=""){
        boolean n = dao.executeUpdate(request.getParameter("textarea"));       //执行创建视图的操作
        if (n){
            out.print("<script>window.alert('视图创建成功！');</script>");
        }else{
            out.print("<script>window.alert('视图创建失败！');</script>");
        }
    }
%>
```

心法领悟 155：视图的妙用。

可通过视图对数据进行查询。例如，一张表中有 30 列，有成千上万行，而用户只需要使用表中的 3 列数据。

便可以为这 3 列创建一个视图，然后在视图中查询所需要的数据，这样会大大提高查询的效率。



实例 156 应用视图查询数据

光盘位置：光盘\MR\06\156

高级

实用指数：★★★★☆

实例说明

视图显示了"伪表"，创建这些表主要是为了以特定的形式显示数据库的内容。使用视图可限制用户访问敏感的数据，帮助用户执行复杂的 SQL 语句。在 Java 程序中实现从视图中查询数据，与从普通表中查询数据的方法是相同的。本实例实现的就是在数据库中创建视图，并通过 Java 程序将视图中的数据查询出来，如图 6.18 所示。

应用视图查询数据

编号	姓名	专业
1	小陈	化工材料
2	小李	数学
3	小张	小学教育
6	小葛	企业管理

图 6.18　应用视图查询数据

关键技术

如果成功地创建了视图，Java 程序可以直接从视图中查询数据，方法与普通表一样。从视图中查询数据的 SQL 语句语法如下：

```
SELECT column_name from viewName
```

参数说明

❶ column_name：从视图中查询的字段。

❷ viewName：要查询的视图名称。

注意：视图是基于表创建的，当表被删除时，基于表创建的视图也就不能再使用。

设计过程

（1）在数据库中定义视图 v_stu，实现从 tb_stu 表中查询数据。具体代码如下：

```
create view v_stu
as
 select * from tb_stu
```

（2）在项目中创建类 StuUtil，在该类中定义操作数据库的相关方法，然后定义从视图中查询数据的方法 getStu()。具体代码如下：

```
public List getStu(){
        List list = new ArrayList<Stu>();                    //定义用于保存返回值的 List 集合
        conn = getConn();                                    //调用获取数据库连接方法
        String sql = "select * from v_stu";                  //定义从视图中查询数据方法
        try {
                Statement statement = conn.createStatement();   //定义 Statement 对象
                ResultSet rest = statement.executeQuery(sql);   //执行查询语句，获取查询结果集
                while(rest.next()){                            //循环遍历查询结果集
                        Stu stu = new Stu();                   //定义与数据表对应的 JavaBean 对象
                        stu.setId(rest.getInt(1));             //应用查询结果集的内容设置 JavaBean 属性
                        stu.setName(rest.getString(2));
                        stu.setSex(rest.getString(3));
                        stu.setSpeciality(rest.getString(4));
                        list.add(stu);                         //向集合中添加对象
                }
        } catch (SQLException e) {
                e.printStackTrace();
        }
        return list;
}
```

心法领悟 156：定义视图的语法规则。

定义视图不能包含以下内容：

COUNT(*)、ROWSET 函数、派生表、自连接、DISTINCT、STDET、VARLANCE、AVG、Float 列、文本列、ntext 列和图像列、子查询、全文谓词（COUTAIN、FREETEXT）。



实例 157　使用视图计算数据

光盘位置：光盘\MR\06\157

中级

实用指数：★★☆

实例说明

利用视图可以简化用户对数据的操作。将一些复杂的操作在视图中进行，可以保证程序的安全。本实例实现的是在视图中计算商品的利润，之后在页面中查询视图，这样就避免了对数据表进行操作。本实例的运行结果如图 6.19 所示。

应用视图查询数据

商品名称	利率
电视机	599.0
Java从入门到精通	14.0
牛仔裤	16.0
沙发	1439.0
女士箱包	14.0
笔记本电脑	2049.0

图 6.19　使用视图计算数据

关键技术

由实例 155 提供的创建视图的语法可知，AS 关键字后是定义视图的一个完整的 SELECT 查询语句。它可以应用多个表。这个子查询还可以包括单行函数和聚合函数、WHERE 子句和 GROUP BY 子句、嵌套的子查询等，但不能包含 ORDER BY 子句。这个子查询的结果将是所创建的视图的内容。

说明： *之所以不允许在视图定义中使用 ORDER BY 子句是为了遵守 ANSI SQL-92 标准。因为对该标准的原理分析需要对结构化查询语言（SQL）的底层结构和它所基于的数学理论进行讨论，因此不能在这里对它进行充分的解释。但是，如果需要在视图中指定 ORDER BY 子句，可以考虑使用以下方法：*

```
USE db_sql
GO
CREATE VIEW ware_v
AS
SELECT TOP 100 PERCENT *
FROM tb_ware14
ORDER BY  售价
GO
```

SQL Server 中引入的 TOP 结构在同 ORDER BY 子句结合使用时是非常有用的。只有在同 TOP 关键词结合使用时，SQL Server 才支持在视图中使用 ORDER BY 子句。

（1）在数据库中创建视图，实现从商品表中查询商品名称和商品利润。具体代码如下：

```
create view v_ware(wName,profit)
as
 select wName,(price - inPrice)as profit from tb_ware
```

（2）在项目中创建类 WareUtil，在该类中定义操作数据库的各种方法，然后定义从视图中查询数据的方法 selectView()，该方法以 List 集合为返回值。具体代码如下：

```
public List selectView() {
    conn = getConn();                                  //获取数据库连接
    Statement cs = null;                               //定义 CallableStatement 对象
    String sql = "Select * from v_ware";               //定义查询视图的 SQL 语句
    List list = new ArrayList();                       //定义保存查询结果的 List 集合
    try {
        cs = conn.createStatement();                   //实例化 Statement 对象
        ResultSet rest = cs.executeQuery(sql);         //执行 SQL 语句
        while (rest.next()) {                          //循环遍历查询结果集
            Ware ware = new Ware();
            ware.setwName(rest.getString(1));
```

```
            ware.setProfit(rest.getFloat(2));
            list.add(ware);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return list;
}
```

（3）在 List.jsp 页面中，以表格的形式把查询结果显示在页面中。具体代码如下：

```
<table width="367" height="66" border="1" align="center">
  <tr>
    <td width="135"><div align="center">商品名称</div></td>
    <td width="216"><div align="center">利率</div></td>
    </tr>
    <%
     List list = (List)request.getAttribute("list");                    //获取保存在 request 对象中的查询结果
     for(int i = 0;i<list.size();i++){                                  //循环遍历查询结果
            Ware ware = (Ware)list.get(i);                              //获取没有 JavaBean 对象
     %>
  <tr>
    <td><div align="center"><%=ware.getwName()%></div></td>            //将查询结果显示在页面中
    <td><div align="center"><%=ware.getProfit()%></div></td>
    </tr>
    <%} %>
 </table>
```

秘笈心法

心法领悟 157：了解视图。

创建视图时，视图的名称存储在 sysobjects 表中。有关视图中所定义的列的信息添加到 syscolumns 表中，而有关视图相关性的信息添加到 sysdepends 表中。另外，CREATE VIEW 语句的文本添加到 syscomments 表中。这与存储过程相似。当首次执行视图时，只有其查询树存储在过程高速缓存中。每次访问视图时，都重新编译其执行计划。

实例 158　使用视图格式化检索出来的数据

光盘位置：光盘\MR\06\158

中级

实用指数：★★☆

实例说明

应用视图有很多好处，首先就是简单，看到的就是需要的；并且在数据库中创建视图后，可以在程序中随意地应用。在本实例中，将首先创建视图，实现从员工表中将员工的姓名、员工的入司时间进行格式化处理，并在程序中将查询结果显示出来，如图 6.20 所示。

使用视图格式化检索出来的数据

员工姓名	入司时间
王宏	2009-8-18
张三	2007-10-7
司马	2007-10-7
张丹	2010-1-10
李玉	2009-09-10

图 6.20　使用视图格式化检索出来的数据

本实例实现的是将入司日期以短日期的格式显示。在 AS 关键字后的 SELECT 子句中应用 CONVERT 函数进行字符转换。

视图不仅可以简化用户对数据的理解，也可以简化对它们的操作。一般情况下，创建和使用视图应遵循以下几点原则。

- 要创建视图，用户必须要有权限。
- 视图必须有唯一的名称。这一点不仅局限于视图与视图之间，并且视图与表之间也不允许拥有相同的名称。
- 创建视图的个数不受限制，用户可基于同一张表创建多个视图。

- ❑ 视图可以嵌套，即可以基于视图创建视图。

设计过程

（1）在数据库中创建视图，实现检索员工表中的数据。具体代码如下：

```
create view v_emp
as
  select name,convert(char(10),enterDate,120) as edate from tb_emp
```

（2）在项目中创建类 EmpDao，在该类中定义操作数据库的方法，然后定义从视图中查询数据的方法 selectView()。具体代码如下：

```
public List selectView() {
    conn = getConn();                                   //获取数据库连接
    Statement cs = null;                                //定义 CallableStatement 对象
    String sql = "Select * from v_emp";                 //定义查询视图的 SQL 语句
    List list = new ArrayList();                        //定义保存查询结果的 List 集合
    try {
        cs = conn.createStatement();                    //执行 SQL 语句
        while (rest.next()) {                           //循环遍历查询结果集
            Emp emp = new Emp();
            emp.setName(rest.getString(1));
            emp.seteDate(rest.getString(2));
            list.add(emp);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return list;
}
```

秘笈心法

心法领悟 158：在视图中 DML 语句遵循的准则。

在一些复杂的视图上的 DML 操作一般遵循以下准则：

- ❑ 不允许违反约束的 DML 操作。
- ❑ 不允许将一个值添加到包含算术表达式的列中。
- ❑ 在非 key_preserved 表上不允许 DML 操作。
- ❑ 在包含组函数、GROUP BY 子句、DISTINCT 关键字或 ROWNUM 的伪视图上不允许 DML 操作。

实例 159	获取数据库中的全部用户视图 光盘位置：光盘\MR\06\159	中级 实用指数：★★☆

实例说明

在某些网站后台或者系统中，要求显示出数据库中所有的用户视图。本实例将介绍如何获取数据库的全部用户视图。运行程序，即可将数据库中的所有用户视图显示出来，如图 6.21 所示。

数据库中的视图
tb_view
tb_Stu_VIEW

图 6.21　获取数据库中的全部用户视图

使用 SQL 语句显示数据库中的所有视图，语法如下：

```
Select * From Sysobjects Where Xtype = 'V' and Status>0
```

参数说明

❶ Sysobjects：该表为系统表，用于存储系统信息。

❷ Xtype：Sysobjects 系统表中的字段，指定了当前记录是何种类型的对象名，类型为 V 表示是视图。

❸ Status：Sysobjects 系统表的状态标识字段，当该字段中的数值大于 0 时表示当前记录所表示的对象为用

户所有，否则为系统所有。

设计过程

（1）创建 UserDao 类，定义数据库连接、更新和关闭的方法。

（2）创建 index.jsp 页面，通过 UserDao 类中的 selectStatic()方法，获取并显示数据库中的所有视图。其关键代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java"
        import="java.sql.*" errorPage=""%>
<jsp:useBean id="dao" scope="request" class="com.pkh.dao.UserDao" />
<%
        ResultSet Rs = dao.selectStatic("Select * From Sysobjects Where Xtype = 'V' and Status>0");          //执行查询语句
        while (Rs.next()){                                                                                   //循环输出结果集
%>
<tr>
        <td bgcolor="#FFFFFF"><div align="center"><%=Rs.getString("Name")%></div></td>
</tr>
<%      }       %>
```

秘笈心法

心法领悟 159：区分动态网页与动态页面的概念。

在 Web 服务器端创建的动态网页与使用客户端脚本程序实现的具有动态视觉效果的网页及 Flash 动画网页是不同的。虽然这两者在浏览器中都可以显示出视觉上的动态效果，并能与用户交互，但这种动态视觉效果是浏览器执行的结果，并不是网页的源文件内容改变后的结果。

实例 160 修改视图

光盘位置：光盘\MR\06\160

中级

实用指数：★★☆

实例说明

本实例演示了如何动态修改一个已创建的视图。运行程序，在如图 6.22 所示页面中的“修改视图”文本框中输入要修改视图的 SQL 语句，单击“修改视图”按钮，如果修改成功，则提示修改成功信息，否则提示修改失败。

图 6.22 修改视图

使用 SQL 语句中的 ALTER VIEW 语句可以修改已创建的视图。语法如下：

```
ALTER VIEW [ < database_name > .] [ < owner > .] view_name [ ( column [ ,...n ] ) ]
[ WITH < view_attribute > [ ,...n ] ]
AS
      select_statement
[ WITH CHECK OPTION ]
```

参数说明

❶ view_name：要修改的视图名称。

❷ column：一列或多列的名称，列名之间用逗号分隔，用于指定视图所显示的列。

设计过程

（1）创建 UserDao 类，定义连接、更新和关闭数据库的方法。其关键代码如下：

```
public class UserDao {
        String url = "jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_database08";
        String username = "sa";
        String password = "";
        private Connection con = null;
```

```
        private Statement stmt = null;
        private ResultSet rs = null;
        public UserDao() {                              //通过构造方法加载数据库驱动
                try {
                        Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
                } catch (Exception ex) {
                        System.out.println("数据库加载失败");
                }
        }
    public boolean Connection() {                       //创建数据库连接
        try {
            con = DriverManager.getConnection(url, username, password);
        } catch (SQLException e) {
            System.out.println(e.getMessage());
            System.out.println("creatConnectionError!");
        }
        return true;
    }
    public boolean executeUpdate(String sql) {          //更新数据库
        if (con == null) {
            Connection();
        }
        try {
            stmt = con.createStatement();
            int iCount = stmt.executeUpdate(sql);
            System.out.println("操作成功，所影响的记录数为" + String.valueOf(iCount));
        } catch (SQLException e) {
            System.out.println(e.getMessage());
            return false;
        }
        closeConnection();
        return true;
    }
}
```

（2）创建 index.jsp 页面，添加 form 表单，设置文本域，执行修改视图的操作。其关键代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java"
        import="java.sql.*" errorPage=""%>
<%
        if ("修改视图".equals(request.getParameter("Submit")) && request.getParameter("textarea")!=""){
                boolean count = dao.executeUpdate((String)request.getParameter("textarea"));
                if (count){
                        out.println("<script type='text/javascript'>window.alert('视图修改成功！');</script>");
                }else{
                        out.println("<script type='text/javascript'>window.alert('视图修改失败！');</script>");
                }
        }
%>
```

秘笈心法

心法领悟 160：理解视图的安全性。

出于视图的安全性考虑，可通过以下方式防止未授权的用户查看特定行或列。

- ❑ 在表中增加一个表示用户名的列。
- ❑ 建立视图，使用户只能看到标有自己用户名的行。
- ❑ 把视图授权给其他用户。

实例 161	删除视图 光盘位置：光盘\MR\06\161	中级 实用指数：★★☆

实例说明

在网站的后台处理中，往往需要程序动态删除数据库中无用的视图，以便节省空间。本实例实现的是动态

删除已创建的视图。运行程序，在如图 6.23 所示页面中选择要删除的视图，然后单击“删除视图”按钮，如果删除成功则提示成功信息，否则提示视图删除失败。

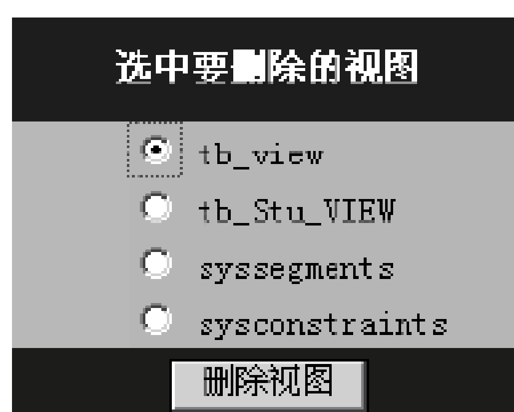


图 6.23　删除视图

关键技术

本实例使用 SQL 语句中的 DROP VIEW 删除已创建的视图。语法如下：

```
DROP VIEW [view name] [,…n]
```

参数说明

view name：要删除的视图名称。

（1）创建 UserDao 类，定义连接、查询、更新和关闭数据库的方法。其关键代码如下：

```
public class UserDao {
        String url = "jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_database08";
        String username = "sa";
        String password = "";
        private Connection con = null;
        private Statement stmt = null;
        private ResultSet rs = null;
        public UserDao() {                                          //通过构造方法加载数据库驱动
                try {
                        Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
                } catch (Exception ex) {
                        System.out.println("数据库加载失败");
                }
        }
      public boolean Connection() {                                 //创建数据库连接
            try {
                  con = DriverManager.getConnection(url, username, password);
            } catch (SQLException e) {
                  System.out.println(e.getMessage());
                  System.out.println("creatConnectionError!");
            }
            return true;
      }
        public ResultSet selectStatic(String sql) throws SQLException {      //对数据库的查询操作
                ResultSet rs=null;
                if (con == null) {
                        Connection();
                }
                try {
                        stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);
                        rs = stmt.executeQuery(sql);
                } catch (SQLException e) {
                        e.printStackTrace();
                }
                closeConnection();
                return rs;
      }
      public boolean executeUpdate(String sql) {                    //更新数据库
            if (con == null) {
                  Connection();
            }
            try {
                  stmt = con.createStatement();
```

```
                int iCount = stmt.executeUpdate(sql);
                System.out.println("操作成功，所影响的记录数为" + String.valueOf(iCount));
            } catch (SQLException e) {
                System.out.println(e.getMessage());
                return false;
            }
            closeConnection();
            return true;
        }
}
```

（2）创建 index.jsp 页面，执行删除视图的语句。其关键代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java"
        import="java.sql.*" errorPage=""%>
<jsp:useBean id="dao" scope="request" class="com.pkh.dao.UserDao" />
<%
        if ("删除视图".equals(request.getParameter("Submit")) && request.getParameter("radiobutton")!=null){
                boolean count = dao.executeUpdate("DROP VIEW "+(String)request.getParameter("radiobutton"));
                if (count){
                        out.println("<script type='text/javascript'>window.alert('视图删除成功！');</script>");
                }else{
                        out.println("<script type='text/javascript'>window.alert('视图删除失败！');</script>");
                }
        }
%>
```

心法领悟 161：理解数据库视图概念。

数据库视图的概念是原始数据库数据的一种变换，是查看表中数据的另外一种方式。可以将视图看成是一个移动的窗口，在其中可以看到感兴趣的数据。视图是从一个或多个实际表中获得的，这些表的数据存放在数据库中。用于产生视图的表叫做视图的基本表。此外，一个视图也可以从另一个视图中产生。

第3篇

图表统计篇

⏭ 第 7 章 JFreeChart 绘图基础

⏭ 第 8 章 基础图表技术

⏭ 第 9 章 扩展图表技术

⏭ 第 10 章 基于 Cewolf 组件的图表编程

第 7 章

JFreeChart 绘图基础

- ⏭ 图表的基础
- ⏭ 设置图表的背景
- ⏭ 处理图表的边框
- ⏭ 修改图表的图例

7.1　图表的基础



实例 162　**基本饼图**　光盘位置：光盘\MR\07\162　**高级**　实用指数：★★★

实例说明

JFreeChart 是一款开源的 Java 图表绘制工具，图表种类丰富，接口通俗易懂，支持多种显示方式，如 Application、Applets、Servlet 和 JSP。本实例将创建一个简单的饼图，其中有 3 个分类，分别为 A、B、C，如图 7.1 所示。

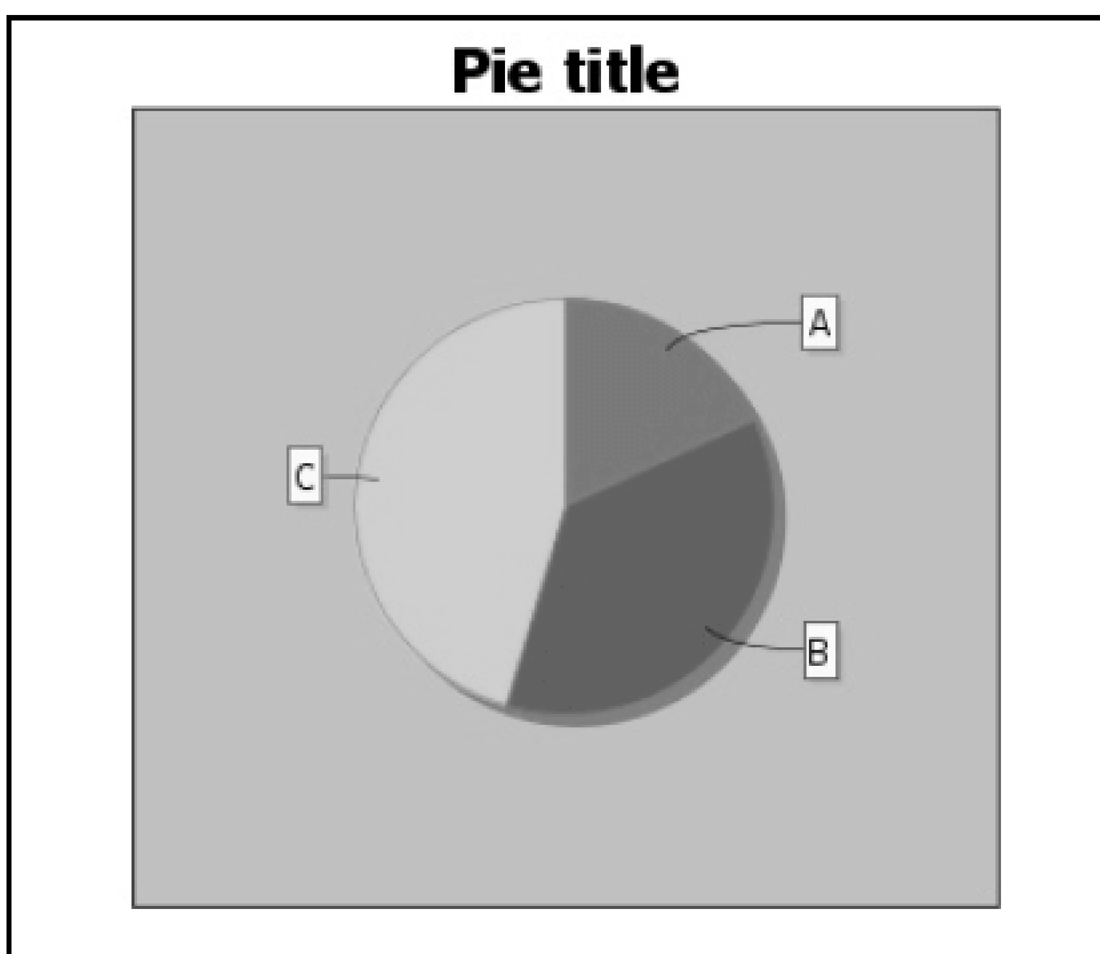


图 7.1　简单饼图

（1）DefaultPieDataset 可以创建饼图的数据集合，使用 setValue()方法可以为数据集合添加数据。语法如下：

```
public void setValue(Comparable key, double value)
```

参数说明

❶ key：要向图表中添加的类别。

❷ value：表示与类别相对应的值。

JFreeChart 把添加到数据集合中的数据以图表的方式显示出来，代码如下：

```
private static PieDataset getPieDataset() {
        //创建一个饼图的数据集
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向饼图的数据集添加数据
        dataset.setValue("A", 200);
        dataset.setValue("B", 400);
        dataset.setValue("C", 500);
        return dataset;
}
```

（2）ChartFactory 是一个图形工厂，其中有很多方法可以把各种数据集合转换成 JFreeChart 对象。例如，使用 createPieChart()方法根据饼图数据集合创建一个 JFreeChart 对象。语法如下：

```
public static JFreeChart createPieChart(String title, PieDataset dataset, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：表示饼图的标题。

❷ dataset：表示饼图的数据集合。

❸ legend：表示是否使用图例。
❹ tooltips：表示是否生成工具栏提示。
❺ urls：表示是否生成 URL 链接。

例如：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        //生成 JFreeChart 对象
        JFreeChart chart = ChartFactory.createPieChart("Pie title",dataset, false, false, false);
        return chart;
}
```

设计过程

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        //创建一个饼图表的数据集
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向饼图表的数据集添加数据
        dataset.setValue("A", 200);
        dataset.setValue("B", 400);
        dataset.setValue("C", 500);
        return dataset;
}
```

（2）使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        //生成 JFreeChart 对象
        JFreeChart chart = ChartFactory.createPieChart("Pie title",dataset, false, false, false);
        return chart;
}
```

（3）JFreeChart 组件提供了一个 Servlet 文件用于获取生成的图表，此 Servlet 文件存储在 JFreeChart 组件包中，所以在使用过程中，需要将其配置到 Web.xml 文件中。其配置方法如下：

```
<servlet>
        <servlet-name>DisplayChart</servlet-name>
        <servlet-class>org.jfree.chart.servlet.DisplayChart</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>DisplayChart</servlet-name>
        <url-pattern>/DisplayChart</url-pattern>
</servlet-mapping>
```

（4）创建 index.jsp 页，用于显示图表。关键代码如下：

```
<body>
        <%
        String filename = ServletUtilities.saveChartAsJPEG(ChartUtil.getJFreeChart(),300,300,session);
        String chartUrl = path+"/DisplayChart?filename="+filename;
      %>
      <center><img alt="" src="<%=chartUrl %>"></center>
</body>
```

心法领悟 162：输出 JFreeChart 生成的图表。

在输出图表之前，首先要生成 JFreeChart 组件所绘制的图表。可通过调用 ServletUtilities 类的 saveChartAsJPEG() 方法进行生成，它将返回一个.jpeg 格式的图表名称。

语法如下：

```
public static String saveChartAsJPEG(JFreeChart chart, int width,int height, HttpSession session) throws IOException
```



实例 163　显示图例　高级

光盘位置：光盘\MR\07\163　实用指数：★★★

实例说明

一般情况下图表都含有图例，JFreeChart 也支持图例的生成。本实例在实例 162 的基础上进行了改动，在饼图中显示图例，如图 7.2 所示。

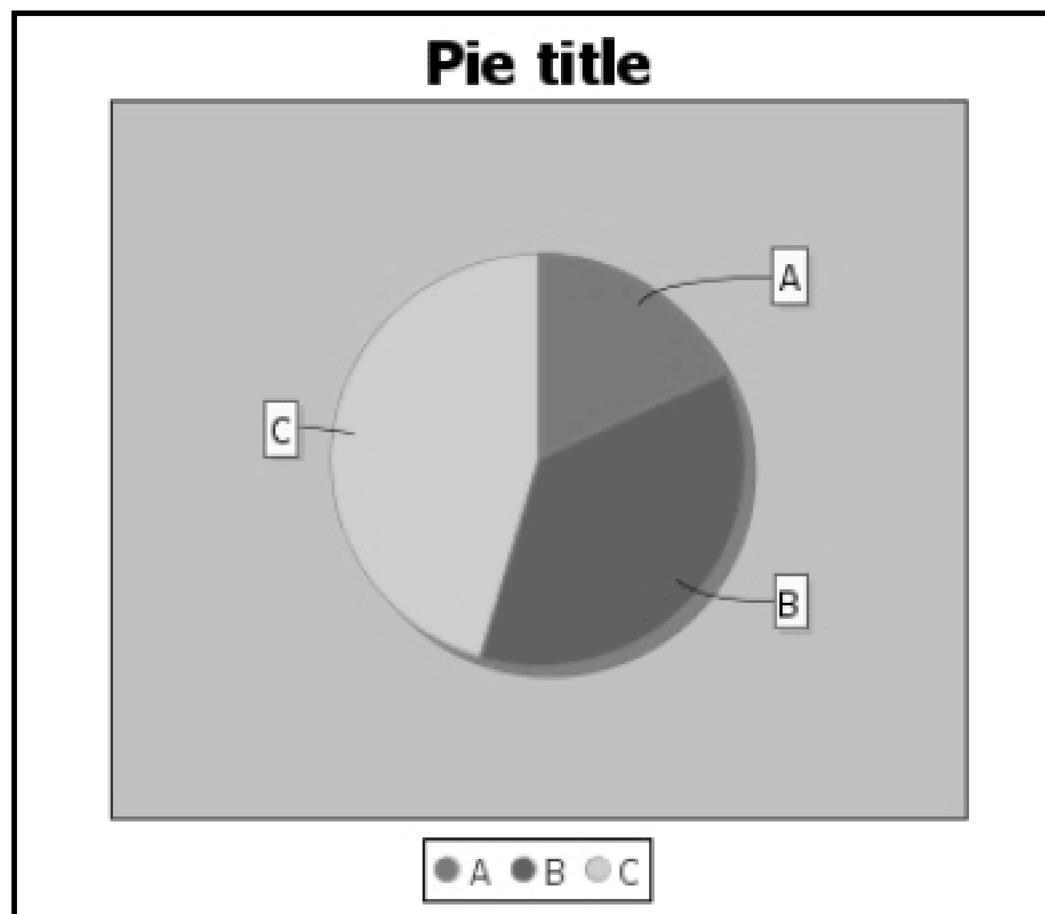


图 7.2　含有图例的饼图

关键技术

在生成 JFreeChart 时可以使用 ChartFactory 的 createPieChart()方法设置图例是否显示。语法如下：

```
public static JFreeChart createPieChart(String title, PieDataset dataset, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：表示饼图的标题。

❷ dataset：表示饼图的数据集合。

❸ legend：表示是否使用图例。

❹ tooltips：表示是否生成工具栏提示。

❺ urls：表示是否生成 URL 链接。

如果将参数 legend 设置为 true，图表中就可以显示出图例。代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        //生成 JFreeChart 对象
        JFreeChart chart = ChartFactory.createPieChart("Pie title",dataset, true, false, false);
        return chart;
}
```

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        //创建一个饼图的数据集
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向饼图的数据集添加数据
        dataset.setValue("A", 200);
        dataset.setValue("B", 400);
        dataset.setValue("C", 500);
        return dataset;
}
```

（2）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，并设置 legend 参数为 true。代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart("Pie title", dataset,true, false, false);
        return chart;
}
```

（3）创建 index.jsp 页，用于显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 163：ChartFactory 生成图表的方法。

ChartFactory 中还有很多类似于 createPieChart(String title，PieDataset dataset，boolean legend，boolean tooltips，boolean urls)的方法可以创建各种图表 ，所有 legend 参数都用来表示是否生成图例。

实例 164　工具栏提示　光盘位置：光盘\MR\07\164　高级　实用指数：★★★

实例说明

不管是在网页还是在应用程序中，有时将鼠标指针放在某一个位置，其附近会弹出一个工具栏提示，能让用户更了解该位置上详细的情况。JFreeChart 也可以为图表生成这种提示，提示的内容是图表上具体的数值和百分比，运行结果如图 7.3 所示。

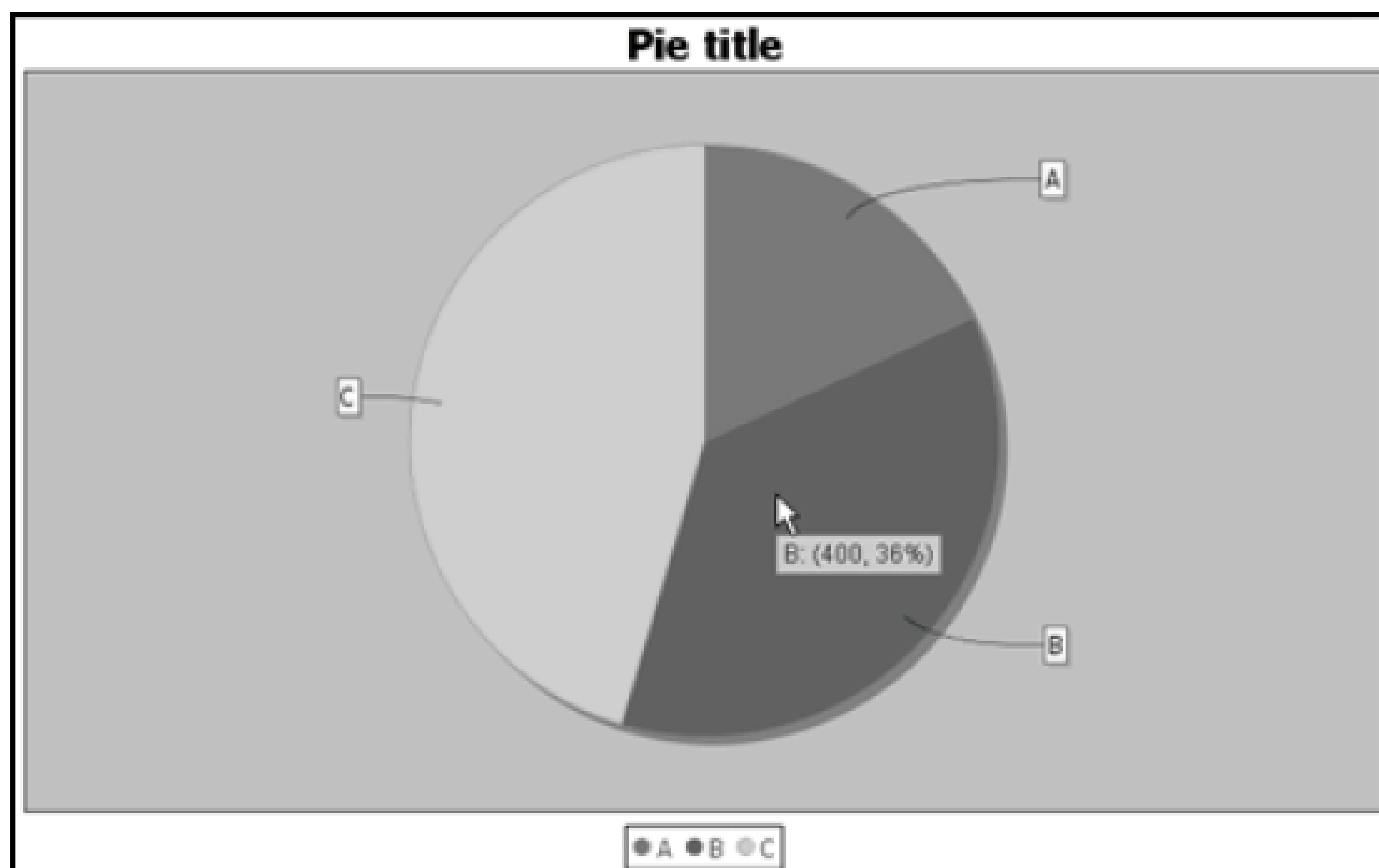


图 7.3　图表工具栏提示

##

在 ChartFactory 类的 createPieChart()方法中可以设置是否显示工具栏的提示。语法如下：

```
public static JFreeChart createPieChart(String title, PieDataset dataset, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：表示饼图的标题。

❷ dataset：表示饼图的数据集合。

❸ legend：表示是否使用图例。

❹ tooltips：表示是否生成工具栏提示。

❺ urls：表示是否生成 URL 链接。

如果将参数 tooltips 设置为 true，则显示工具栏提示；如果设置为 false，则不显示。代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        //生成 JFreeChart 对象
        JFreeChart chart = ChartFactory.createPieChart("Pie title", dataset,true, true, false);
```

```
        return chart;
}
```

设计过程

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        //创建一个饼图的数据集
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向饼图的数据集添加数据
        dataset.setValue("A", 200);
        dataset.setValue("B", 400);
        dataset.setValue("C", 500);
        return dataset;
}
```

（2）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，并设置 tooltips 参数为 true。代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        //生成 JFreeChart 对象
        JFreeChart chart = ChartFactory.createPieChart("Pie title", dataset,true, true, false);
        return chart;
}
```

（3）创建 inex.jsp 页，用于显示 JFreeChart 图表。具体代码参见配书光盘。

秘笈心法

心法领悟 164：ChartFactory 中显示工具栏提示的方法。

在 ChartFactory 中，createPieChart(String title,PieDataset dataset,boolean legend,boolean tooltips,boolean urls)方法用于创建饼图，参数 tooltips 为 true 时即可显示工具栏提示。ChartFactory 中的其他图表，如柱形图、线形图、区域图等都有 tooltips 参数，都可以用来控制工具栏提示是否启用。

实例 165	乱码问题 光盘位置：光盘\MR\07\165	高级 实用指数：★★★★

JFreeChart 是国外的开源框架，图表中使用的默认字体是 SansSerif，在生成图表时如果使用汉字，会产生乱码，如图 7.4 所示。

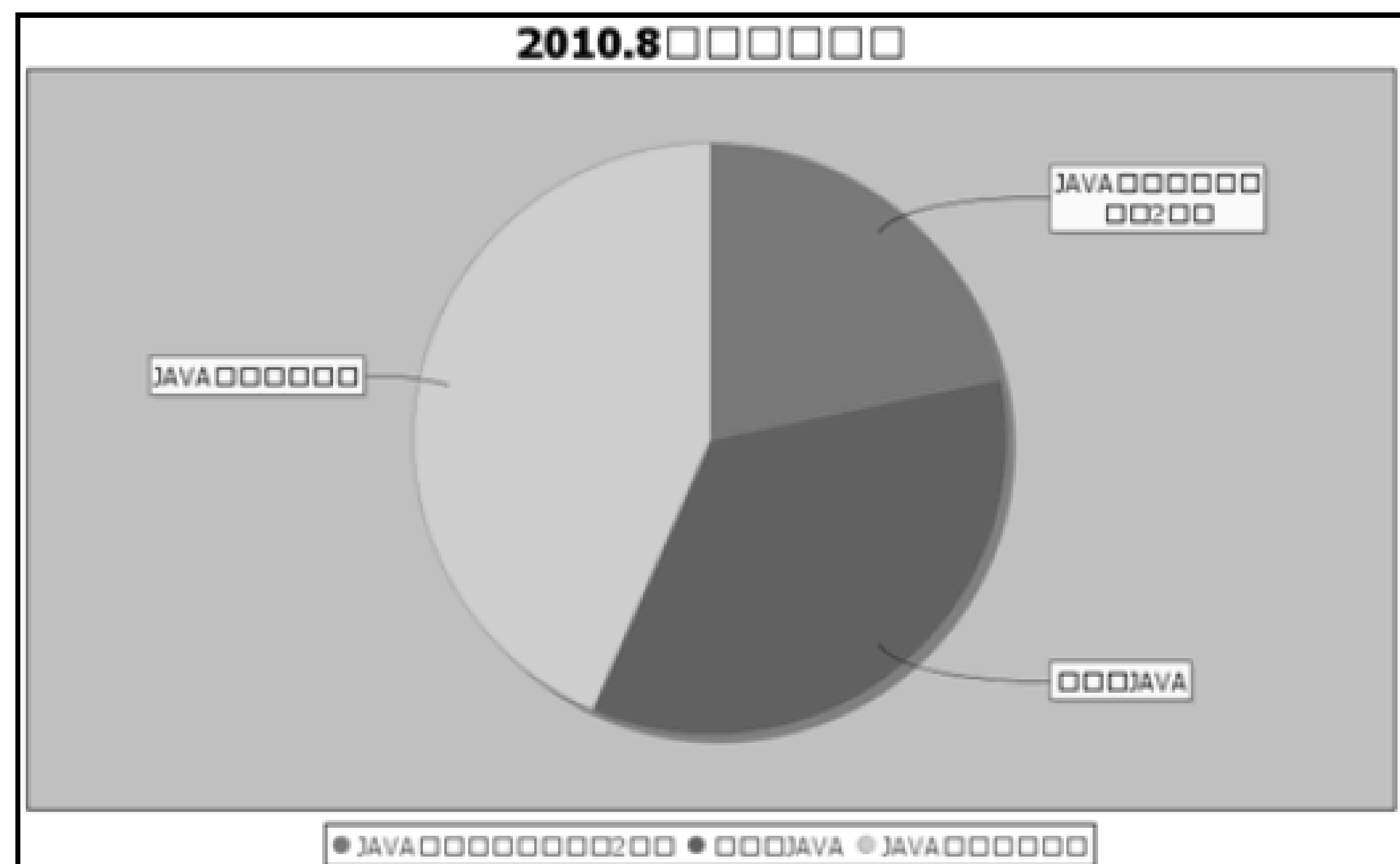


图 7.4　饼图中的乱码

如果希望在 JFreeChart 中使用汉字，需要重新设置汉字的字体，如“宋体”、“黑体”、“楷体”等。本实例的饼图中有 3 处需要输入汉字，分别为图表标题、图表本身和图例，将这 3 处的字体都设置为“宋体”，

运行效果如图 7.5 所示。

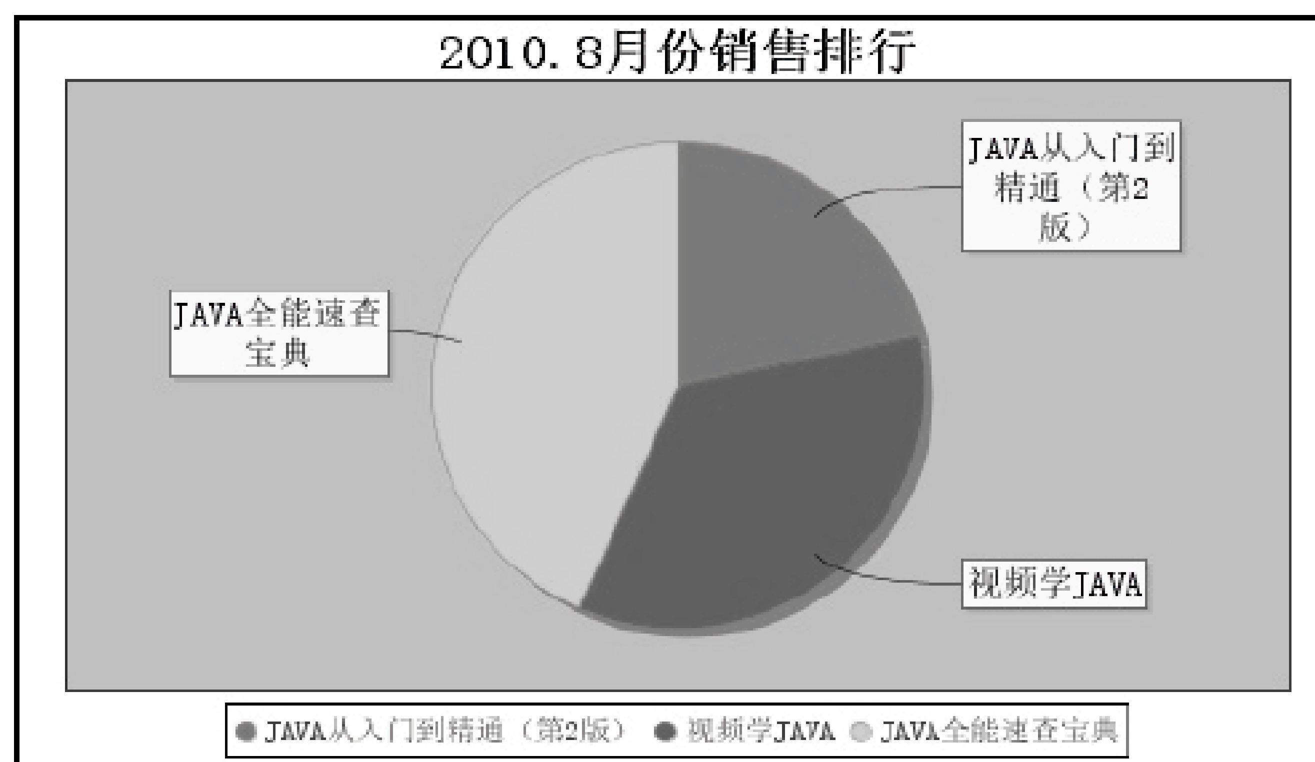


图 7.5　饼图中的汉字

关键技术

（1）使用 JFreeChart 的 getPlot()方法，可以获取 PiePlot 对象。语法如下：

```
public Plot getPlot()
```

（2）使用 PiePlot 的 setLabelFont()方法可以设置图表本身的字体。语法如下：

```
public void setLabelFont(Font font)
```

参数说明

font：表示图表的字体。

（3）使用 JFreeChart 的 getTitle()方法，可以获取 TextTitle 对象。语法如下：

```
public TextTitle getTitle()
```

（4）使用 TextTitle 的 setFont()方法，可以设置图表标题的字体。语法如下：

```
public void setFont(Font font)
```

参数说明

font：表示图表标题的字体。

（5）使用 JFreeChart 的 getLegend()方法，可以获取图例。语法如下：

```
public LegendTitle getLegend()
```

（6）使用 LegendTitler 的 setItemFont(Font font)方法，可以设置图例的字体。语法如下：

```
public void setItemFont(Font font)
```

参数说明

font：表示图表中图例的字体。

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
	//创建一个饼图的数据集
	DefaultPieDataset dataset = new DefaultPieDataset();
	//向饼图的数据集添加数据
	dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
	dataset.setValue("视频学 JAVA", 800);
	dataset.setValue("JAVA 全能速查宝典", 1000);
	return dataset;
}
```

（2）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
	//获取数据集
	PieDataset dataset = getPieDataset();
	//生成 JFreeChart 对象
	JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
	return chart;
}
```

（3）创建 setPiePoltFont()方法，用于设置图表、图表标题和图例的字体。代码如下：

```
public static void setPiePoltFont(JFreeChart chart) {
	//图表（饼图）
```

```
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置图表字体
        piePlot.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.BOLD, 20));
        //图例
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 12));
}
```

（4）创建 index.jsp 页，用于显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 165：设置字体样式。

设置字体时，使用了 new Font("宋体", Font.PLAIN, 12)构造字体对象。其中，“宋体”表示要设置的字体名称；12 表示字体的大小；Font.PLAIN 是一个常量，表示字体使用普通的样式，该常量还可以使用 Font.BOLD 和 Font.ITALIC 来代替，分别表示粗体和斜体。

实例 166	显示数值 光盘位置：光盘\MR\07\166	高级 实用指数：★★★

实例说明

一般图表中都会显示数值，可能是具体的值，也可能是百分比。在 JFreeChart 的图表中，对数值的显示很方便，显示方式也非常灵活。本实例把《JAVA 从入门到精通（第 2 版）》、《视频学 JAVA》和《JAVA 全能速查宝典》的 8 月份销售数据统计生成饼图，在其中显示出具体销售的数值，如图 7.6 所示。

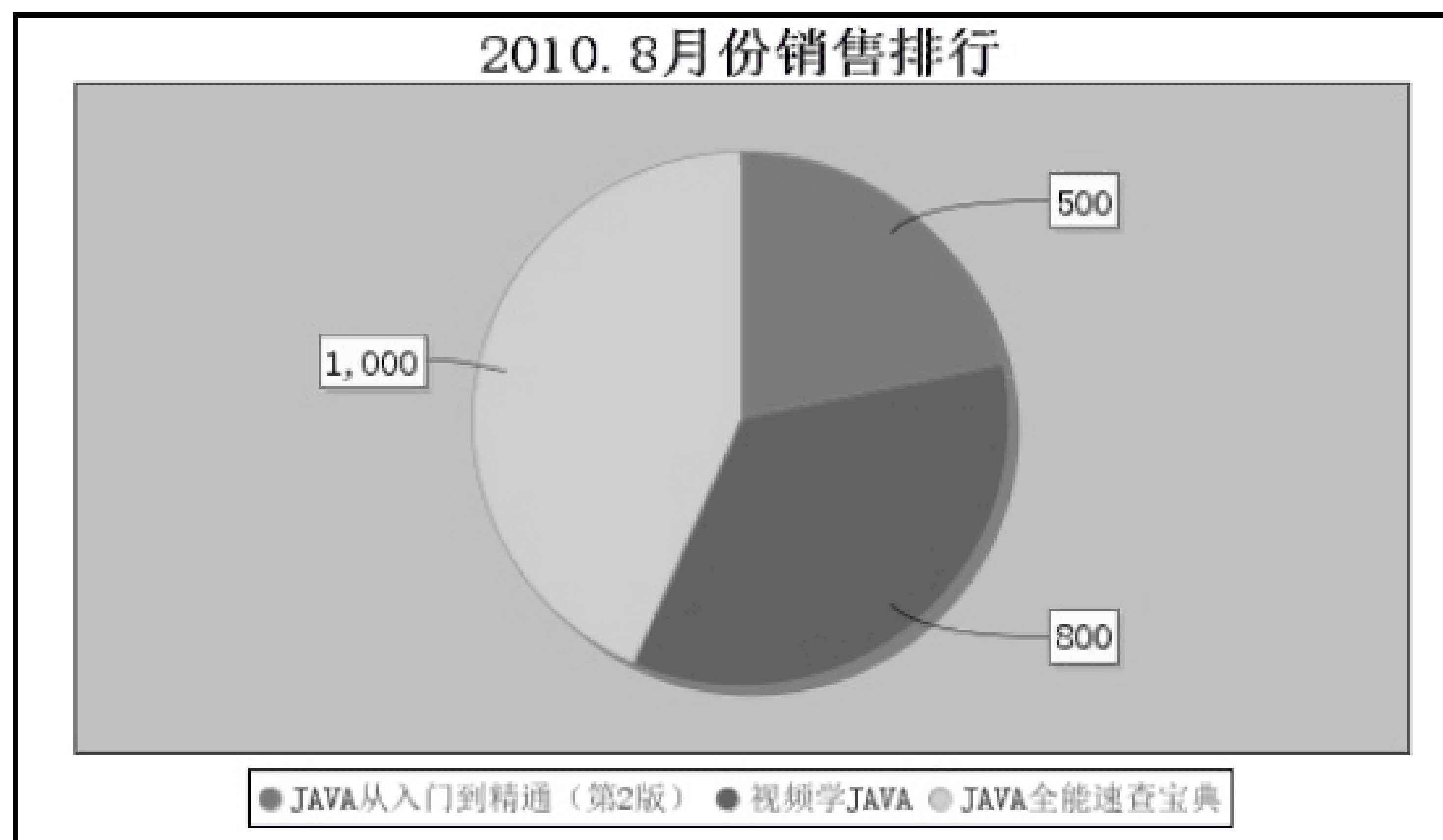


图 7.6　含有数值的图表

在默认情况下，JFreeChart 只在图表中显示类别名称而不显示数值。使用 StandardPieSectionLabelGenerator 的构造函数可以重新生成图表标签。语法如下：

```
public StandardPieSectionLabelGenerator(String labelFormat)
```

参数说明

labelFormat：表示标签显示的样式。其中：

- ❑ “{0}”就是 JFreeChart 使用的默认值，使用它则表示图表中显示类别名称。
- ❑ “{1}”表示图表中显示类别的具体数值。
- ❑ “{2}”表示图表中要显示当前类别在总数中的百分比。
- ❑ “{3}”表示图表中所有类别相加的总值。

设计过程

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        //创建一个饼图的数据集
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向饼图的数据集添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        return dataset;
}
```

（2）ChartFactory()类根据饼图的数据集合创建一个 JFreeChart 对象，代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        //生成 JFreeChart 对象
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        return chart;
}
```

（3）创建 setPiePoltFont()方法，用于设置图表、图表标题和图例的字体。代码如下：

```
public static void setPiePoltFont(JFreeChart chart) {
        //图表（饼图）
        PiePlot piePlot = (PiePlot) chart.getPlot();
        piePlot.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.BOLD, 20));
        //图例
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 12));
}
```

（4）创建 setPiePoltNum(JFreeChart chart)方法修改 JFreeChart 图表显示为数值。代码如下：

```
public static void setPiePoltNum(JFreeChart chart) {
        //图表（饼图）
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置饼图标签显示
        piePlot.setLabelGenerator(new StandardPieSectionLabelGenerator("{1}"));
}
```

（5）创建 index.jsp 页，用于显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 166：StandardPieSectionLabelGenerator 构造函数的参数含义。

在使用 StandardPieSectionLabelGenerator 构造函数重写 JFreeChart 的标签时，构造函数的字符串参数中可以填写{0}、{1}、{2}、{3}，这些有格式化功能的字符都有其含义。这个构造函数的字符串参数使用方法并不止这些，把刚才格式化的字符组合起来，可以产生不同的效果。如希望显示“类别名称：类别数值”，可以表示为“{0}：{1}”；希望显示“类别百分比：类别总额”，可以表示为“{2}：{3}”；希望显示“类别数值（单位）”，可以表示为“{1}本”。

实例 167	抗锯齿设置 光盘位置：光盘\MR\07\167	高级 实用指数：★★

实例说明

图像在显示时，由于受到显示器分辨率的制约，物体周围会出现三角形的锯齿状。抗锯齿即指对图像边缘

进行柔化处理，使其更平滑。JFreeChart 默认都是打开抗锯齿设置，如果关闭抗锯齿设置，图像边缘会显示得参差不齐。本实例实现的是关闭抗锯齿设置，运行结果如图 7.7 所示。

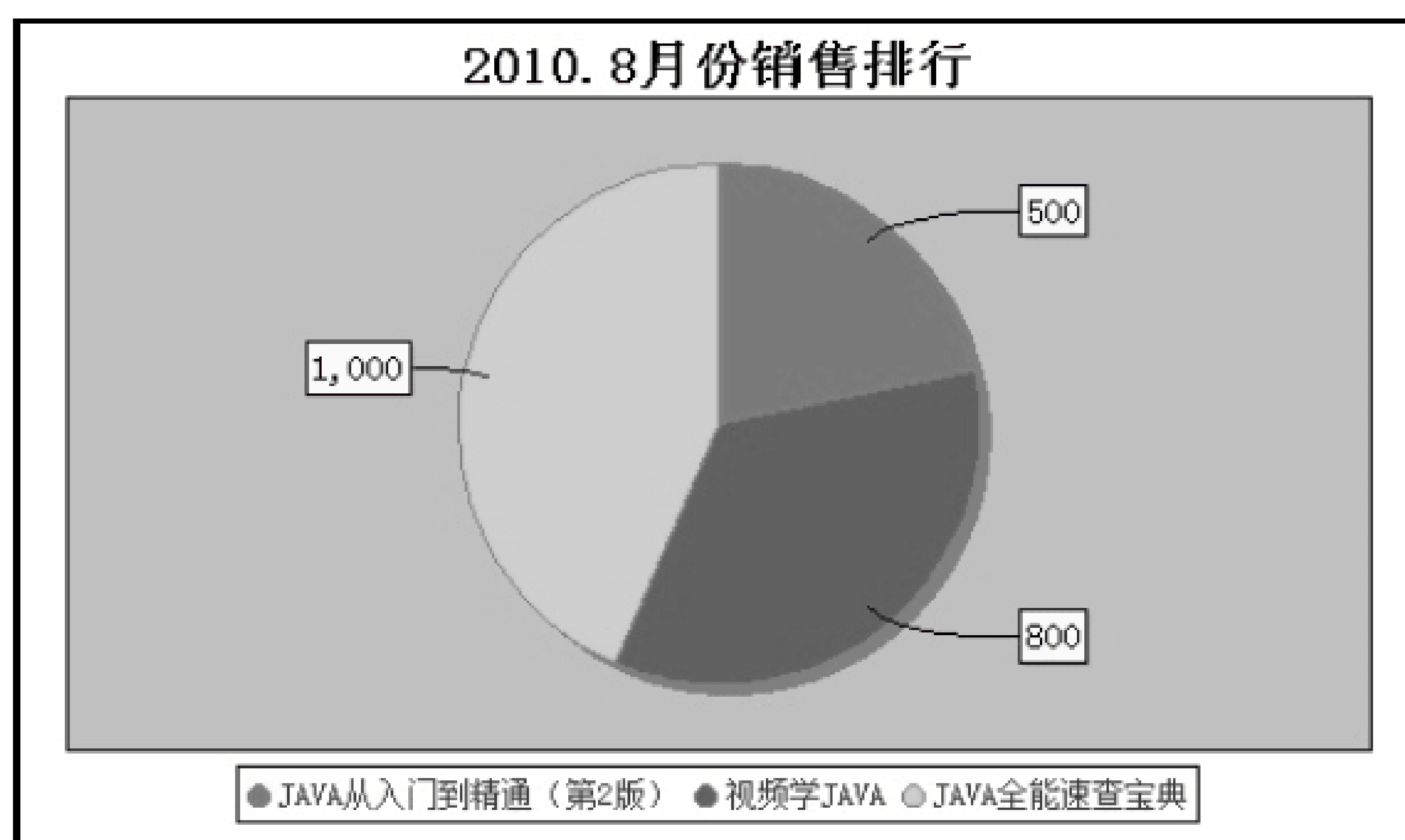


图 7.7　不使用抗锯齿设置

关键技术

获取 JFreeChart 对象后，可以使用 JFreeChart 类的 setAntiAlias (boolean b)方法指定是否设置抗锯齿。语法如下：

```
public void setAntiAlias(boolean flag)
```

参数说明

flag：是否设置抗锯齿。当 flag 值为 true 时，表示打开抗锯齿，图表各颜色边缘会显示得比较平滑；当 flag 值为 false 时，表示关闭抗锯齿设置，图表各颜色边缘会显示出锯齿。

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
	//创建一个饼图的数据集
	DefaultPieDataset dataset = new DefaultPieDataset();
	//向饼图的数据集添加数据
	dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
	dataset.setValue("视频学 JAVA", 800);
	dataset.setValue("JAVA 全能速查宝典", 1000);
	return dataset;
}
```

（2）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias()为 false，关闭抗锯齿。代码如下：

```
public static JFreeChart getJFreeChart() {
	//获取数据集
	PieDataset dataset = getPieDataset();
	JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
	//关闭抗锯齿
	chart.setAntiAlias(false);
	return chart;
}
```

（3）创建 setPiePoltFont()方法，用于设置图表、图表标题和图例的字体。代码如下：

```
public static void setPiePoltFont(JFreeChart chart) {
	//图表（饼图）
	PiePlot piePlot = (PiePlot) chart.getPlot();
	piePlot.setLabelFont(new Font("宋体", Font.PLAIN, 14));
	//标题
	TextTitle textTitle = chart.getTitle();
	textTitle.setFont(new Font("宋体", Font.BOLD, 20));
	//图示
	LegendTitle legendTitle = chart.getLegend();
```

```
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 12));
}
```

（4）创建 index.jsp 页，用于显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 167：图表的抗锯齿。

抗锯齿是一种图像处理技术，先根据图像物体周围进行采点，再根据采点情况进行处理。不同的抗锯齿技术，其采点的方式和方法，以及处理图像的方法都不一样。使用抗锯齿技术虽然可以让图像更美观，但是经过一系列的技术运算，自然会消耗一部分系统资源。

7.2 设置图表的背景

实例 168	设置背景图 光盘位置：光盘\MR\07\168	高级 实用指数：★★★★

实例说明

JFreeChart 的默认背景为灰色，为了使图表在显示时更美观一些，可以为其添加一个背景图。本例将演示如何设置背景图，运行结果如图 7.8 所示。

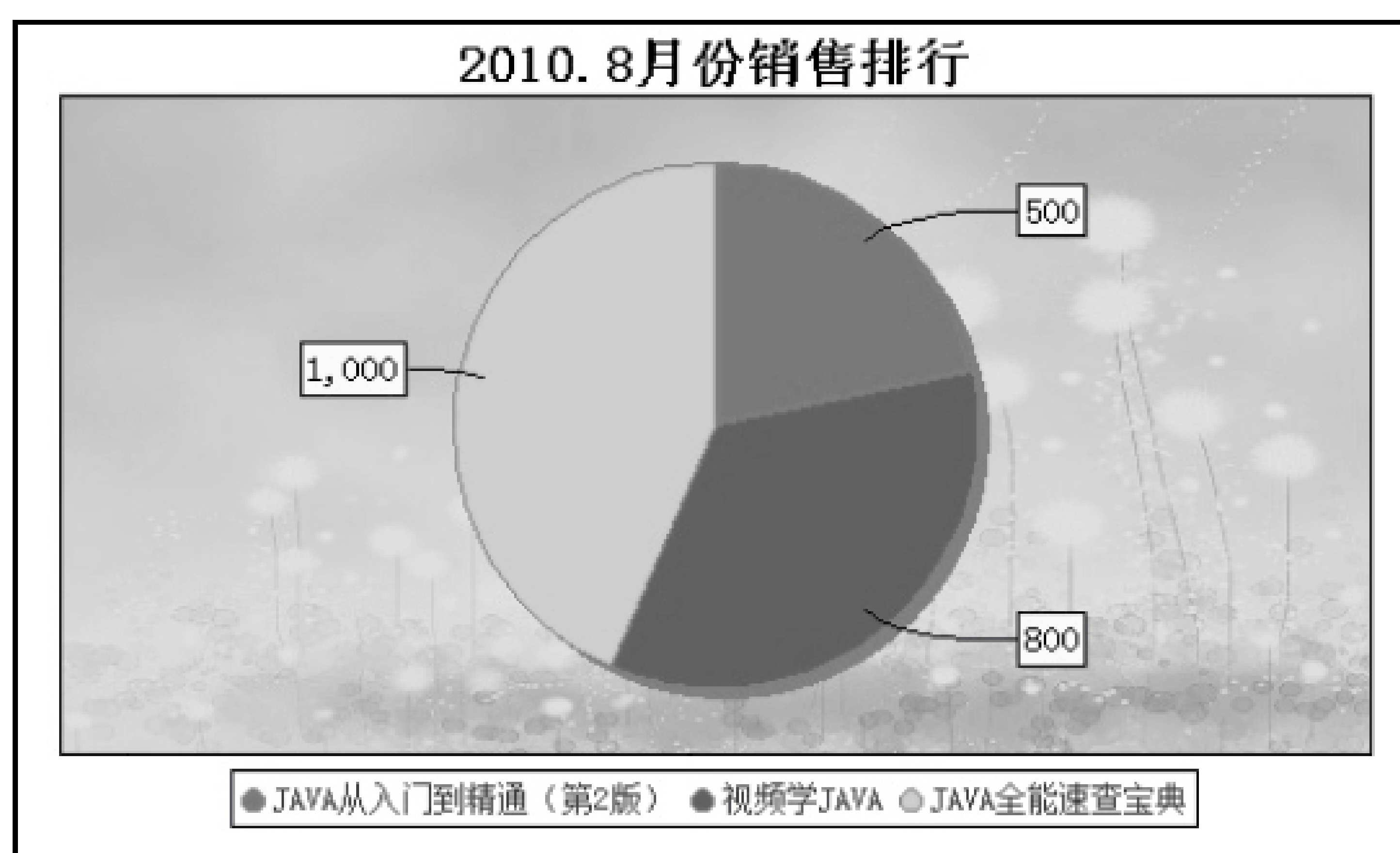


图 7.8　设置图表背景图

关键技术

使用 PiePlot 类的 setBackgroundImage()方法可以设置图表的背景图。语法如下：

```
public void setBackgroundImage(Image image)
```

参数说明

image：用于设置图表的背景图。

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        //创建一个饼图的数据集
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向饼图的数据集添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        return dataset;
}
```

（2）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 SetAntiAlias() 为 false，关闭抗锯齿。代码如下：

```
public static JFreeChart getJFreeChart() {
	//获取数据集
	PieDataset dataset = getPieDataset();
	JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
	//关闭抗锯齿
	chart.setAntiAlias(false);
	return chart;
}
```

（3）创建 setBackgroundImage (JFreeChart chart,String imgPath)方法，用于设置背景图。代码如下：

```
public static void setBackgroundImage(JFreeChart chart,String imgPath) {
	Image image = null;
	try {
		//读取图片
		image = ImageIO.read(new FileInputStream(imgPath));
	} catch (IOException e) {
		e.printStackTrace();
	}
	PiePlot piePlot = (PiePlot) chart.getPlot();
	//设置饼图背景图
	piePlot.setBackgroundImage(image);
}
```

（4）创建 index.jsp 页，显示 JFreeChart 生成的带背景图的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 168：读取图片的方法。

本实例在读取图片时，应用的是 javax.imageio.ImageIO 类的 read(InputStream stream)方法，该方法可以从文件输入流中读取图片数据，然后返回一个 java.awt.Image 对象。

实例 169	设置背景图片透明度 光盘位置：光盘\MR\07\169	高级 实用指数：★★★★

实例说明

为 JFreeChart 设置背景图时，对图片的透明度可以进行调整。本实例将演示如何设置饼图背景图的透明度，运行结果如图 7.9 所示。

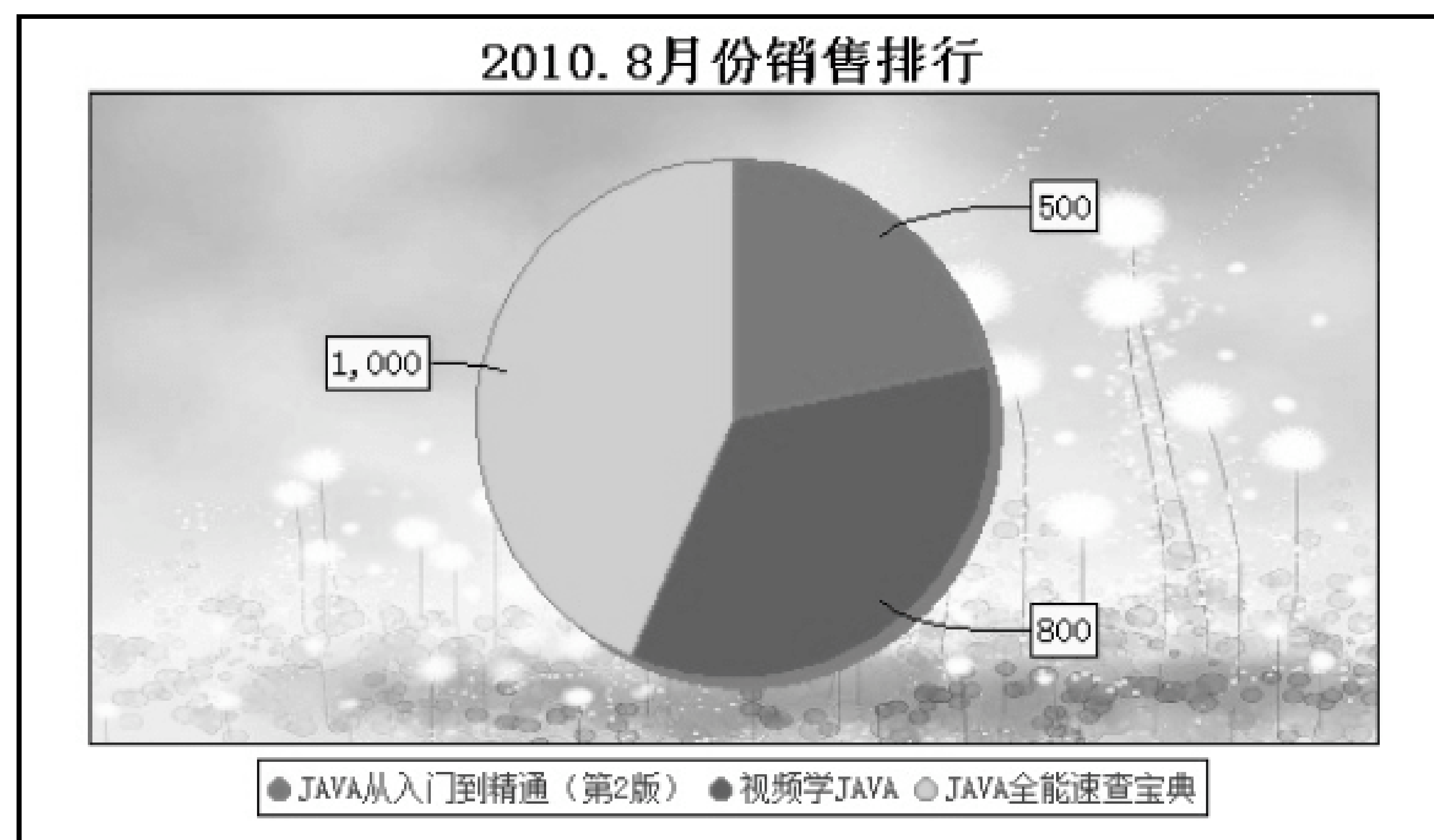


图 7.9　设置图表背景图透明度

使用 PiePlot 的 setBackgroundImageAlpha()方法可以设置背景图的透明度。语法如下：

```
public void setBackgroundImageAlpha(float alpha)
```

参数说明

alpha：用于设置背景图的透明度。如果没有背景图，该设置是不起作用的。参数 alpha 的值位于 0～1 之间，float 类型。值越小，透明度越小，背景图越模糊；反之，值越大，透明度越大，背景图越清晰。

设计过程

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
	//创建一个饼图的数据集
	DefaultPieDataset dataset = new DefaultPieDataset();
	//向饼图的数据集添加数据
	dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
	dataset.setValue("视频学 JAVA", 800);
	dataset.setValue("JAVA 全能速查宝典", 1000);
	return dataset;
}
```

（2）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias()为 false，关闭抗锯齿。代码如下：

```
public static JFreeChart getJFreeChart() {
	//获取数据集
	PieDataset dataset = getPieDataset();
	JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
	//关闭抗锯齿
	chart.setAntiAlias(false);
	return chart;
}
```

（3）创建 setPiePoltFont()方法，用于设置图表、图表标题和图例的字体。代码如下：

```
public static void setPiePoltFont(JFreeChart chart) {
	//图表（饼图）
	PiePlot piePlot = (PiePlot) chart.getPlot();
	//设置图表字体
	piePlot.setLabelFont(new Font("宋体", Font.PLAIN, 14));
	//标题
	TextTitle textTitle = chart.getTitle();
	textTitle.setFont(new Font("宋体", Font.BOLD, 20));
	//图例
	LegendTitle legendTitle = chart.getLegend();
	legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 12));
}
```

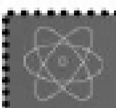
（4）创建 setBackgroundImage (JFreeChart chart)方法设置背景图，同时设置背景图的透明度为 1。代码如下：

```
public static void setBackgroundImage(JFreeChart chart) {
	Image image = null;
	try {
		image = ImageIO.read(new File("backgroundImage.jpg"));
	} catch (IOException e) {
		e.printStackTrace();
	}
	PiePlot piePlot = (PiePlot) chart.getPlot();
	piePlot.setBackgroundImage(image);
	//设置背景透明度
	piePlot.setBackgroundImageAlpha(1f);
}
```

（5）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

心法领悟 169：图表背景的透明度。

实例中设置背景图的透明度为 1，表示背景图达到清晰的效果；如果透明度为 0，背景图将无法显示，只能显示图表的背景色。



实例 170　设置背景色　高级

光盘位置：光盘\MR\07\170　实用指数：★★★★

实例说明

JFreeChart 可以为图表设置背景色，使其看上去更美观。可以单独使用，也可以与背景图和背景图片的透明度组合起来共同使用。本实例将图表的背景色设置为橙色，运行结果如图 7.10 所示。

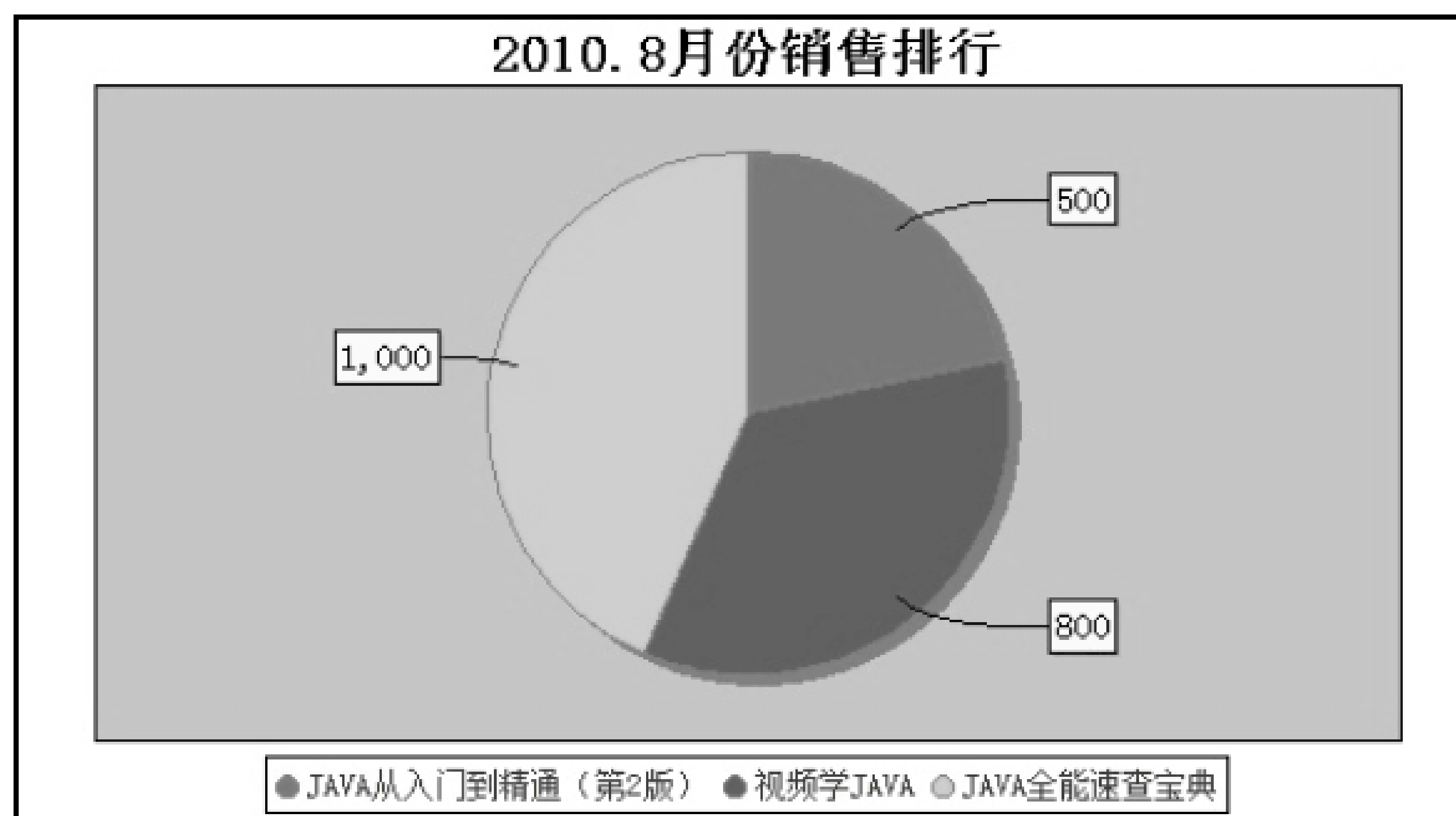


图 7.10　设置图表背景色

关键技术

使用 PiePlot 的 setBackgroundPaint()方法可以为饼图设置背景色。语法如下：

```
public void setBackgroundPaint(Paint paint)
```

参数说明

paint：表示要设置的背景颜色值。

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        //创建一个饼图的数据集
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向饼图的数据集添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        return dataset;
}
```

（2）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 SetAntiAlias()为 false，关闭抗锯齿。代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        //关闭抗锯齿
        chart.setAntiAlias(false);
        return chart;
}
```

（3）创建 setBackgroundColor(JFreeChart chart)方法，修改 JFreeChart 的默认颜色，使用 Color.orange 设置图表背景色为橙色。代码如下：

```
public static void setBackgroundColor(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
```

```
		//设置饼图背景色
		piePlot.setBackgroundPaint(Color.orange);
}
```

（4）创建 index.jsp 页，用于显示生成的带背景色的 JFreeChart 图表。具体代码参见配书光盘。

秘笈心法

心法领悟 170：应用 Color 类设置不同的颜色。

在 Color 类中定义的常量只为用户提供了一些基本的颜色，如果希望使用更丰富的颜色，可以使用 Color 的构造方法直接填写色值，如 new Color(200,220,202)、new Color(200,220,202,255)等。

7.3 处理图表的边框

实例 171	隐藏图表边框 光盘位置：光盘\MR\07\171	高级 实用指数：★

实例说明

JFreeChart 在显示图表时默认都会显示一个边框，边框可以把图表圈在内部，使其与标题和图例分开。本实例实现的是把图表的边框取消，同时将图表背景色和窗体背景色都设置为白色，使二者看起来像是一个整体，运行结果如图 7.11 所示。



图 7.11　隐藏边框

关键技术

使用 PiePlot 的 setOutlineVisible()方法可以设置是否显示饼图边框，语法如下：

```
public void setOutlineVisible(boolean visible)
```

参数说明

visible：表示是否显示饼图边框。如果设置为 true，表示显示边框；如果设置为 false，则表示不显示边框。

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
		//创建一个饼图的数据集
		DefaultPieDataset dataset = new DefaultPieDataset();
		//向饼图的数据集添加数据
```

```
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        return dataset;
}
```

（2）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias() 为 false，关闭抗锯齿。代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        //关闭抗锯齿
        chart.setAntiAlias(false);
        return chart;
}
```

（3）创建 setOutline()方法取消饼图边框，同时设置饼图背景色为白色，与窗体背景色一致。代码如下：

```
public static void setOutline(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
        piePlot.setBackgroundPaint(Color.white);
        //取消边框
        piePlot.setOutlineVisible(false);
}
```

（4）创建 index.jsp 页，显示生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 171：显示图表的边框。

JFreeChart 的图表边框默认状态下是显示的，本实例中如果只是把窗体背景色调成白色，则可以非常清晰地看到图表边框，如图 7.12 所示。



图 7.12　显示边框

实例 172	图表边框颜色和笔触 光盘位置：光盘\MR\07\172	高级 实用指数：★★★



图表边框的宽度默认很细，通过设置其笔触可以调节边框的宽度。同样，用户也可以根据实际需要为图表边框设置自己喜欢的颜色。本实例实现的是把饼图边框宽度调粗，饼图的背景色和窗体背景色设置为白色，边框设置为橙色，如图 7.13 所示。

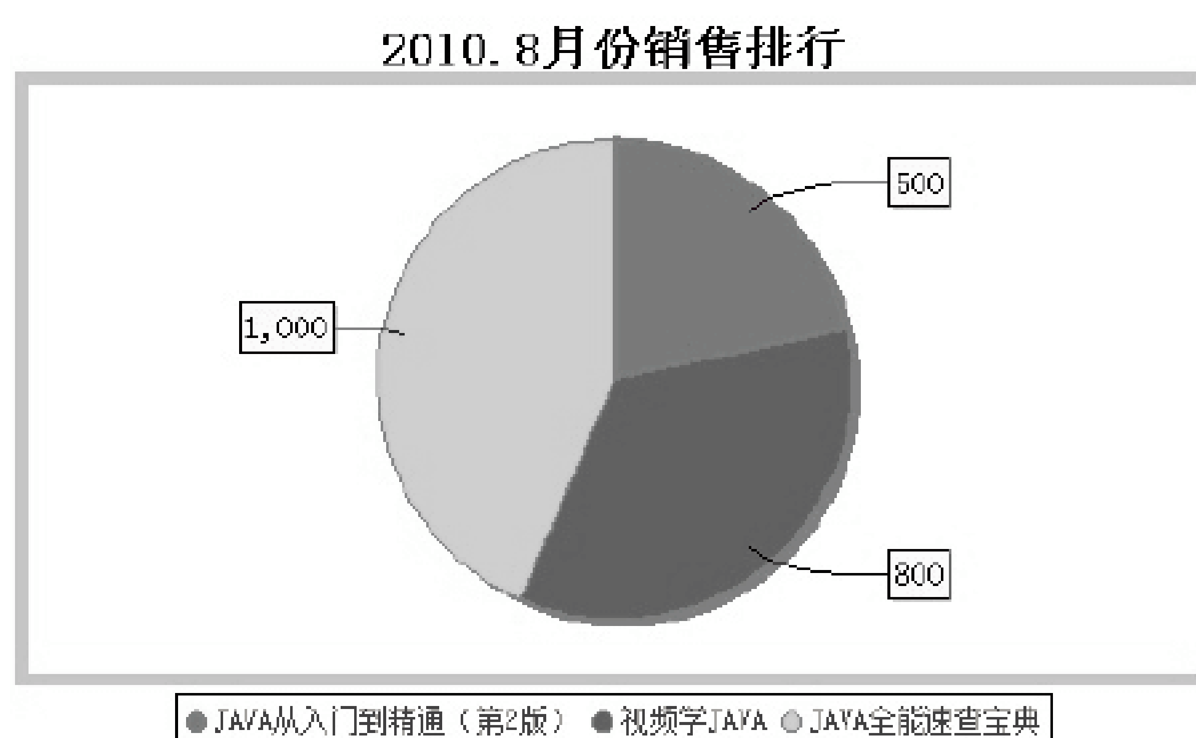


图 7.13　设置饼图边框颜色和笔触

关键技术

（1）使用 PiePlot 类的 setOutlineStroke()方法可以为边框设置笔触。语法如下：

```
public void setOutlineStroke(Stroke stroke)
```

参数说明

stroke：表示边框的笔触，可以用 BasicStroke 进行实例化。

（2）使用 PiePlot 类的 setOutlinePaint()方法可以为边框设置颜色。语法如下：

```
public void setOutlinePaint(Paint paint)
```

参数说明

paint：表示边框的颜色，可以使用 Paint 的实现类——Color 类或者其常量来实现。

示例代码如下：

```
public void setOutline(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
        piePlot.setBackgroundPaint(Color.white);
        Stroke stroke = new BasicStroke(5);
        //设置边框笔触
        piePlot.setOutlineStroke(stroke);
        //设置边框颜色
        piePlot.setOutlinePaint(Color.orange);
}
```

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        //创建一个饼图的数据集
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向饼图的数据集添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        return dataset;
}
```

（2）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias()为 false，关闭抗锯齿。代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        //关闭抗锯齿
        chart.setAntiAlias(false);
        return chart;
}
```

（3）创建 setOutline()方法，使用 BaseStroke 的构造方法实例化 Stroke，并且设置笔触的宽度为 5；再使用 setOutlineStroke()方法设置边框笔触；然后使用 PiePlot 的 setOutlinePaint()方法设置边框为橙色。代码如下：

```
public static void setOutline(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
        piePlot.setBackgroundPaint(Color.white);
        Stroke stroke = new BasicStroke(5);
        //设置边框笔触
        piePlot.setOutlineStroke(stroke);
        //设置边框颜色
        piePlot.setOutlinePaint(Color.orange);
}
```

（4）创建 index.jsp 页，显示生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 172：利用图表的颜色隐藏图表边框。

本实例中图表边框为橙色，图表背景色和网页背景色都为白色。如果把图表边框的颜色设置成与图表背景色或者窗体背景色一致，可以达到隐藏图表边框的效果。

7.4　修改图表的图例



实例 173　设置图例背景色　　高级

光盘位置：光盘\MR\07\173　　实用指数：★★

实例说明

JFreeChart 的图例背景色默认为白色，也可根据实际需要对其进行调整。本实例中将图例的背景色设置为橙色，运行结果如图 7.14 所示。

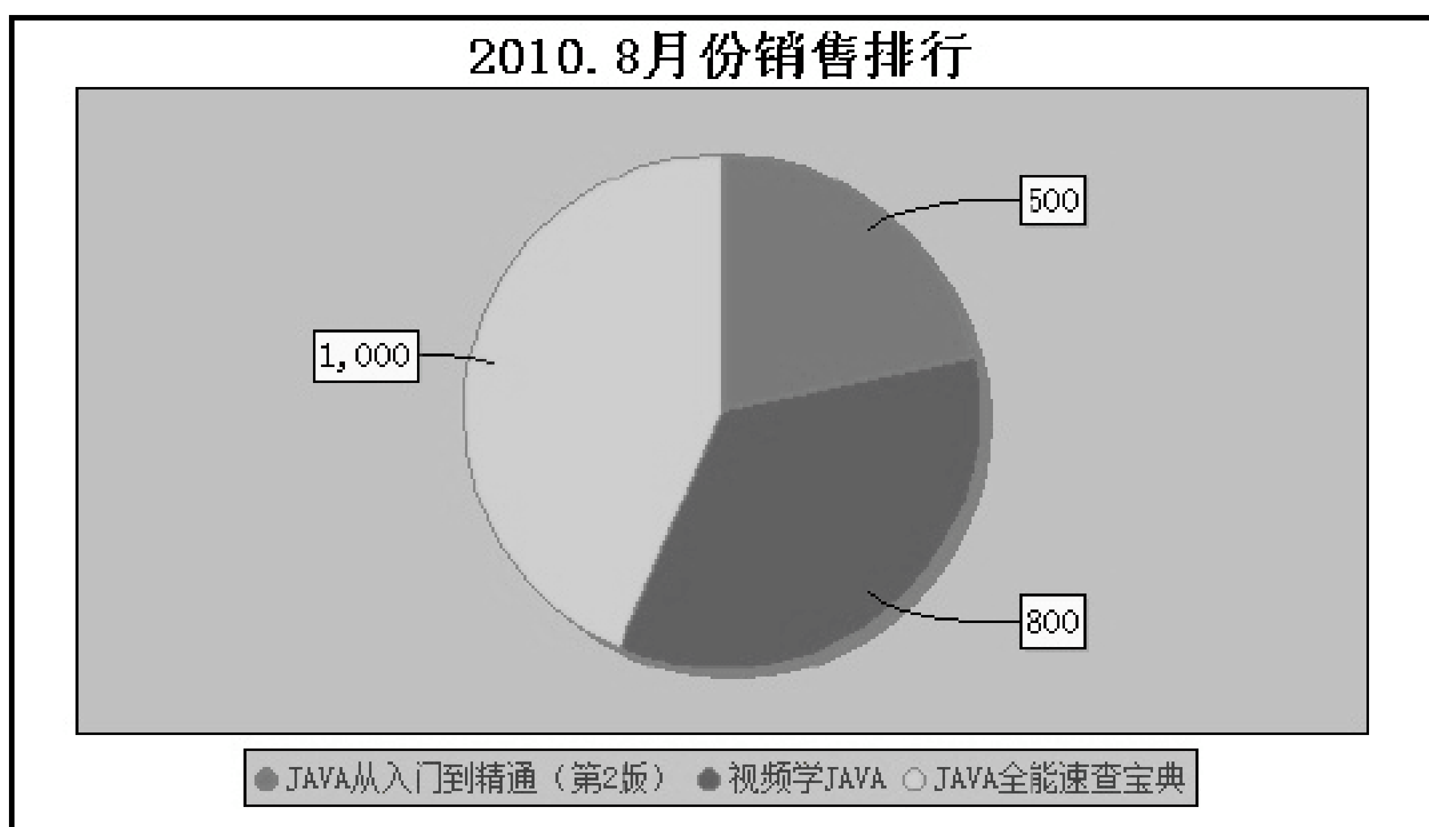


图 7.14　设置图例背景色

关键技术

在 JFreeChart 中，LegendTitle 类用于处理图例。使用 LegendTitle 类的 setBackgroundPaint()方法可以设置图例的背景色。语法如下：

```
public void setBackgroundPaint(Paint paint)
```

参数说明

paint：表示图例的背景色。

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        //创建一个饼图的数据集
```

```
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向饼图的数据集添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        return dataset;
}
```

（2）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias() 为 false，关闭抗锯齿。代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        //关闭抗锯齿
        chart.setAntiAlias(false);
        return chart;
}
```

（3）创建 setLegendTitle()方法，使用 LegendTitle 的 setBackgroundPaint()方法设置图例的背景色为橙色。代码如下：

```
public static void setLegendTitle(JFreeChart chart) {
        LegendTitle legendTitle = chart.getLegend();
        //设置图例背景色
        legendTitle.setBackgroundPaint(Color.orange);
}
```

（4）创建 index.jsp 页，显示生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 173：图例背景色。

在设置图例背景色时，不要把背景色与图例中的颜色设置为相同，以免无法区分。

实例 174	设置图例边框 光盘位置：光盘\MR\07\174	高级 实用指数：★★

实例说明

JFreeChart 图例的边框与图表边框不同，主要通过设置边框的距离来确定边框的粗细。本实例把饼图图例的边框去掉，运行结果如图 7.15 所示。

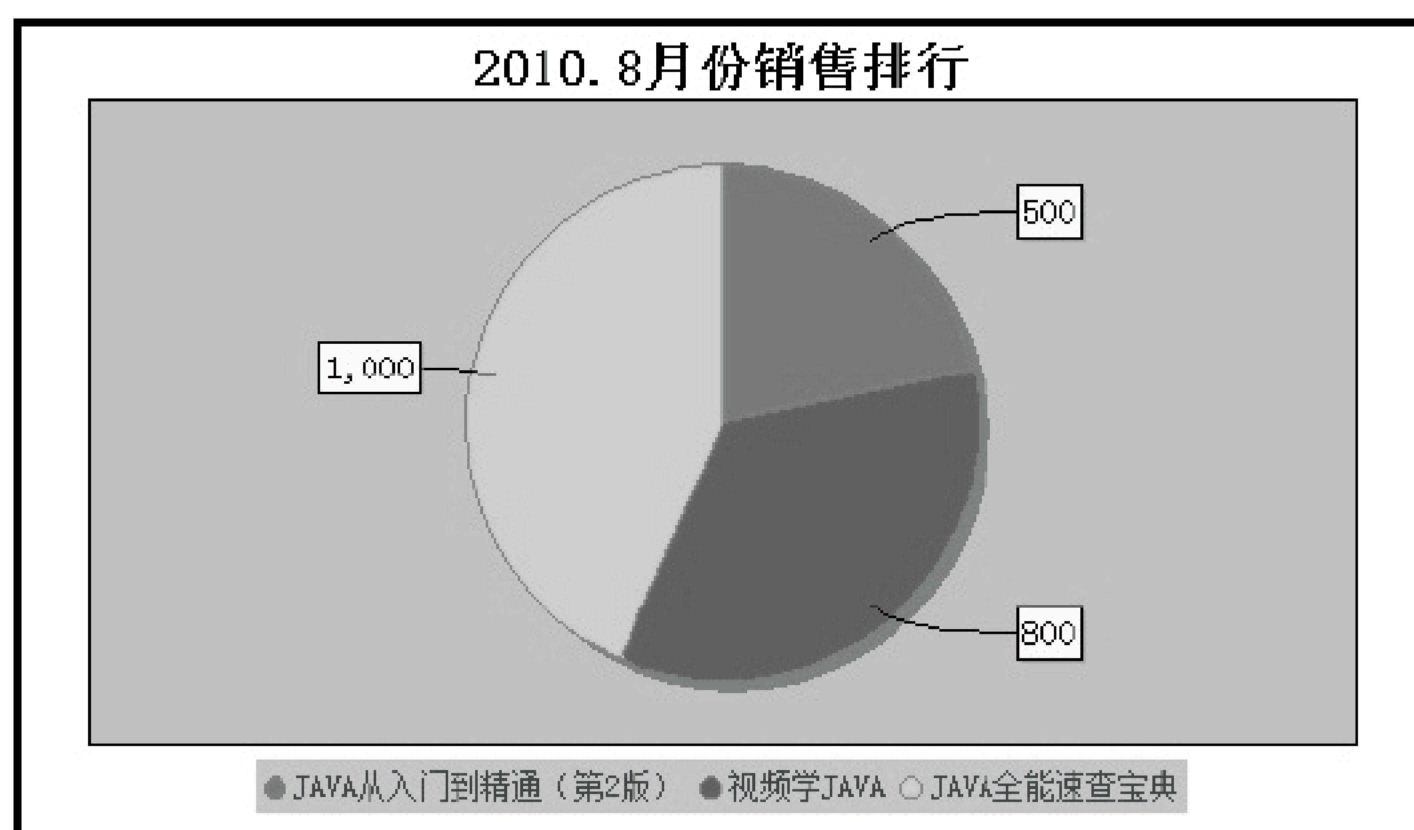


图 7.15　设置图例边框

使用 LegendTitle 的 setBorder()方法可以设置图例的边框。语法如下：

```
public void setBorder(double top, double left, double bottom,double right)
```

参数说明

❶ top：表示图例上边的宽度。

❷ left：表示图例左边的宽度。

❸ bottom：表示图例底边的宽度。

❹ right：表示图例右边的宽度。

使用方法如下：

```
public void setLegendTitle(JFreeChart chart) {
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setBackgroundPaint(Color.orange);
        //设置图例边框
        legendTitle.setBorder(0, 0, 0, 0);
}
```

设计过程

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        //创建一个饼图的数据集
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向饼图的数据集添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        return dataset;
}
```

（2）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias()为 false，关闭抗锯齿。代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        //关闭抗锯齿
        chart.setAntiAlias(false);
        return chart;
}
```

（3）创建 setLegendTitle()方法，使用 LegendTitle 的 setBackgroundPaint()方法设置图例的背景色为橙色，同时设置图例的边框为 0。代码如下：

```
public void setLegendTitle(JFreeChart chart) {
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setBackgroundPaint(Color.orange);
        //设置图示边框
        legendTitle.setBorder(0, 0, 0, 0);
}
```

（4）创建 index.jsp 页，显示生成的图表，具体代码请参见配书光盘。

秘笈心法

心法领悟 174：应用 setBorder(BlockBorder border)方法设置图示的边框。

应用 BlockBorder 对象可以设置图示的边框，只要在创建 BlockBorder 对象时，在其构造方法中传递图示四周的边框值，然后将 BlockBorder 对象作为 setBorder()方法的参数即可。



实例 175	设置图例边缘间距 光盘位置：光盘\MR\07\175	高级 实用指数：★★★

实例说明

图例的边缘与边框不同，它确定的是整个图例（包括边框）与周边的距离。本实例设置图例与左边间距为

10，与上边间距为 0，与右边间距为 246，与下边间距为 10，运行结果如图 7.16 所示。

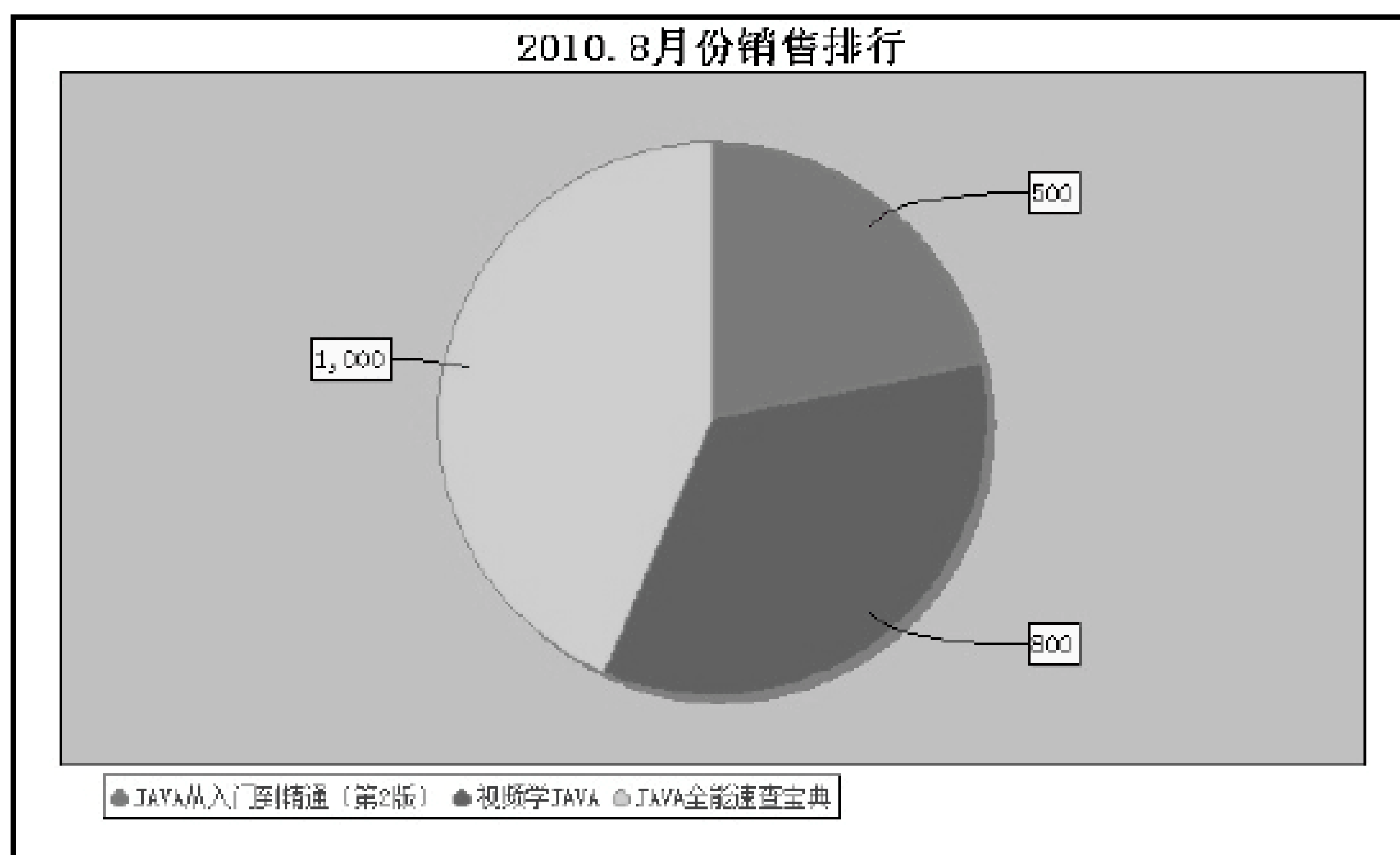


图 7.16　设置图例边缘

■ 关键技术

LegendTitle 的 setMargin()方法用于设置图例边缘的间距，语法如下：

```
public void setMargin(double top, double left, double bottom,double right)
```

参数说明

❶ top：表示图例上边的距离。

❷ left：表示图例左边的距离。

❸ bottom：表示图例底边的距离。

❹ right：表示图例右边的距离。

使用方法如下：

```
public void setLegendTitle(JFreeChart chart) {
        LegendTitle legendTitle = chart.getLegend();
        //设置图例间距
        legendTitle.setMargin(0, 10, 10, 246);
}
```

■

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合，代码如下：

```
private static PieDataset getPieDataset() {
        //创建一个饼图的数据集
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向饼图的数据集添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        return dataset;
}
```

（2）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias()为 false，关闭抗锯齿。代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        //关闭抗锯齿
        chart.setAntiAlias(false);
        return chart;
}
```

（3）创建 setLegendTitle()方法，使用 LegendTitle 的 setMargin()方法设置图例与四边的距离。代码如下：

```
public static void setLegendTitle(JFreeChart chart) {
        LegendTitle legendTitle = chart.getLegend();
        //设置图示间距
```

```
        legendTitle.setMargin(0, 10, 10, 246);
}
```

（4）创建 index.jsp 页，显示生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 175：移动图例的位置。

可以通过设置图例的边缘间距来移动图例的位置，但根据图表所在的网页布局来看，不管怎么移动图例都只能显示在图表下方。



实例 176　**设置图例字体颜色**　高级

光盘位置：光盘\MR\07\176　实用指数：★★★

实例说明

在 JFreeChart 中，图例的字体颜色默认是黑色，也可根据需要对字体颜色进行调整。本实例将图例的字体调整为紫色，运行结果如图 7.17 所示。

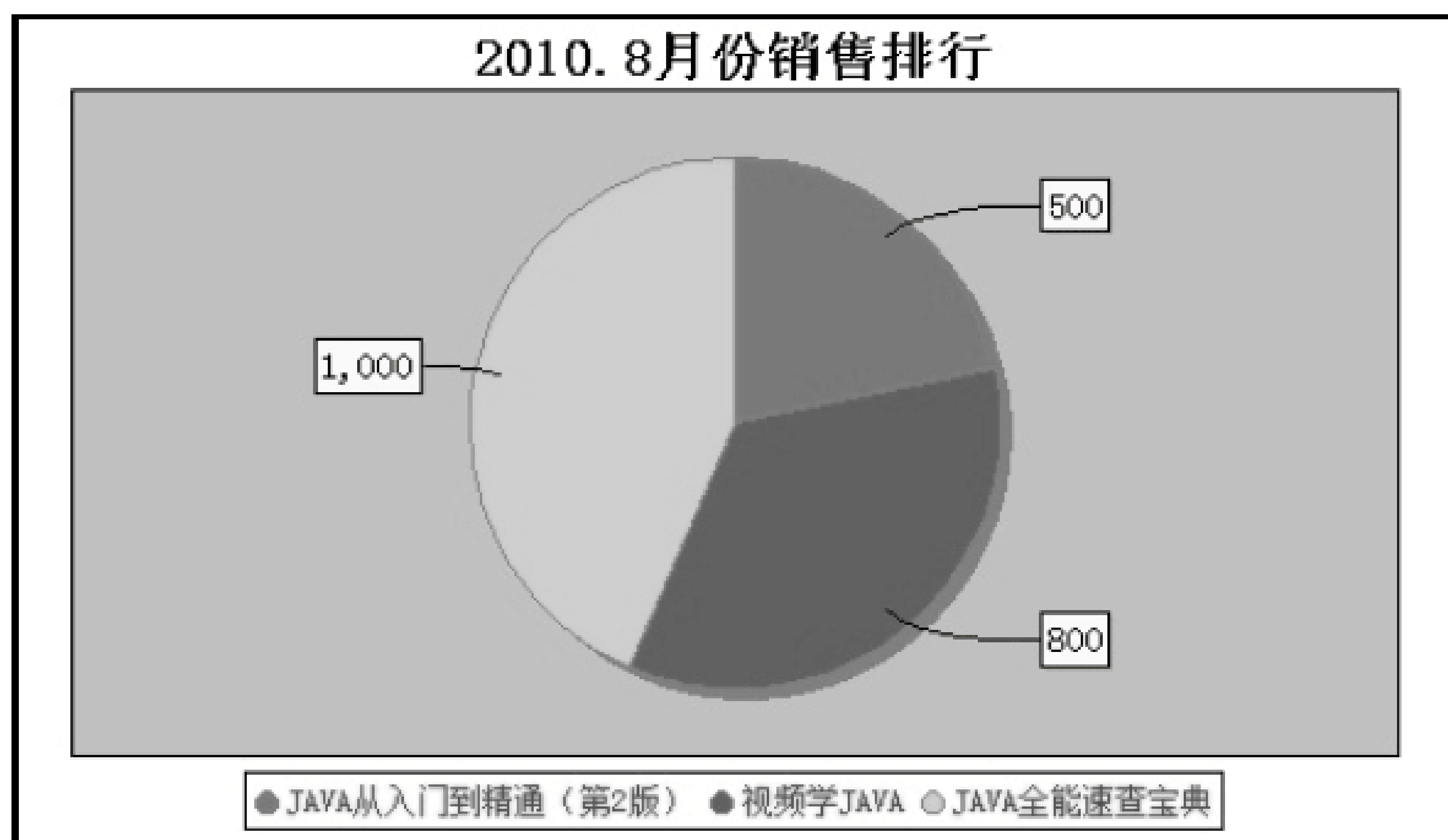


图 7.17　设置图例字体颜色

关键技术

使用 LegendTitle 的 setItemPaint()方法可以为图例设置字体颜色，语法如下：

```
public void setItemPaint(Paint paint)
```

参数说明

paint：表示图例字体的颜色。

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合，代码如下：

```
private static PieDataset getPieDataset() {
        //创建一个饼图的数据集
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向饼图的数据集添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        return dataset;
}
```

（2）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias() 为 false，关闭抗锯齿。代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
```

```
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        //关闭抗锯齿
        chart.setAntiAlias(false);
        return chart;
}
```

（3）创建 setLegendTitle()方法，使用 LegendTitle 的 setItemPaint()方法设置图例字体的颜色为紫色。

```
public static void setLegendTitle(JFreeChart chart) {
        LegendTitle legendTitle = chart.getLegend();
        //设置图示颜色
        legendTitle.setItemPaint(Color.MAGENTA);
}
```

（4）创建 index.jsp 页，显示生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 176：注意图例的字体颜色。

图例的字体颜色可以根据实际需要进行设置。在设置字体颜色时需要注意，不能与图例的背景色相同或者相近，否则分辨不出图例中的文字。

实例 177	设置图例位置 光盘位置：光盘\MR\07\177	高级 实用指数：★★★

实例说明

图例的默认位置在图表下方，也可根据实际需要将其调整到其他位置。本实例把图例位置调整到图表右侧，运行结果如图 7.18 所示。

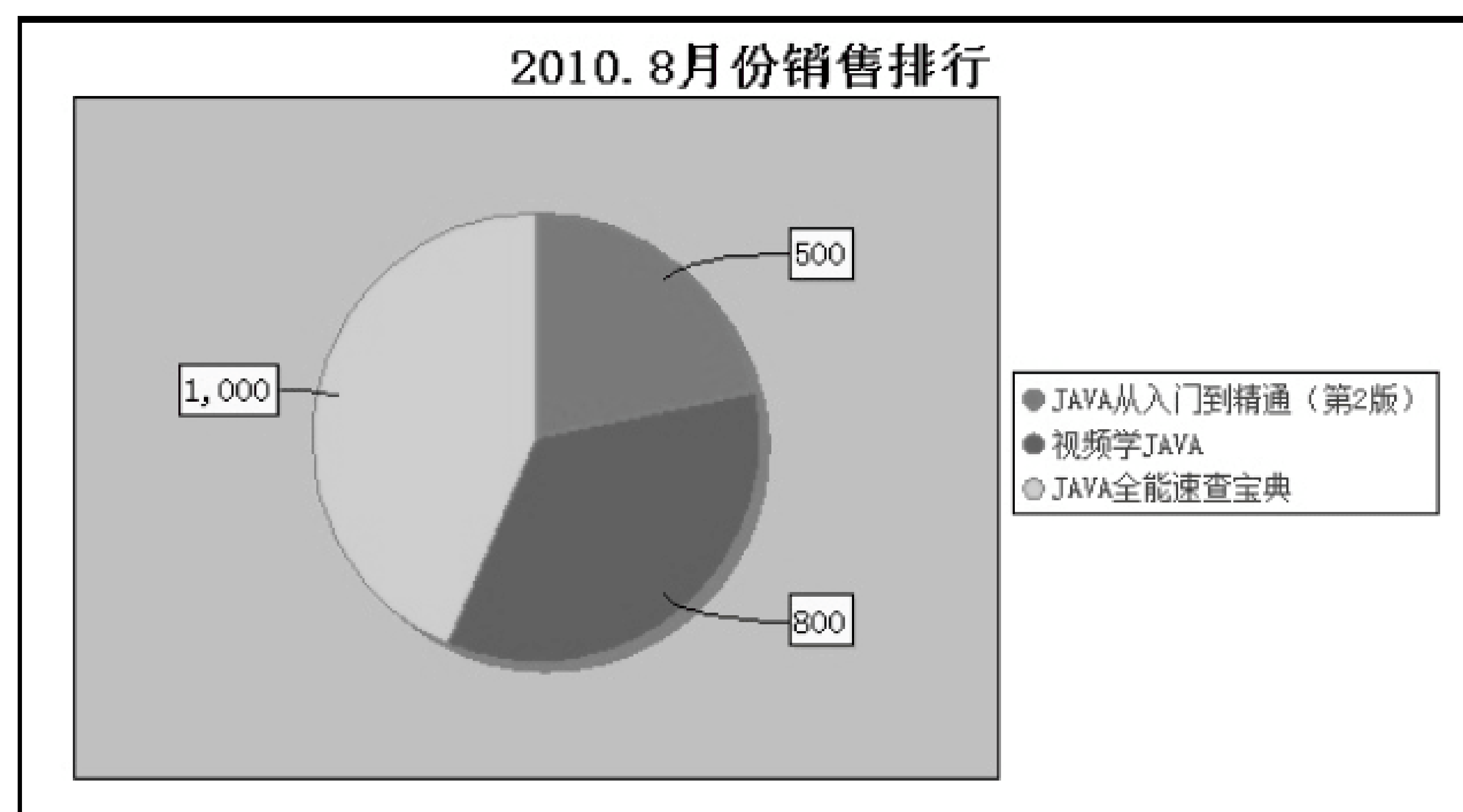


图 7.18　设置图例位置

关键技术

使用 LegendTitle 的 setPosition()方法可以设置图例在图表中的位置，语法如下：

```
public void setPosition(RectangleEdge position)
```

参数说明

position：表示图例在图表中的位置。

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        //创建一个饼图表的数据集
        DefaultPieDataset dataset = new DefaultPieDataset();
```

```
		//向饼图表的数据集添加数据
		dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
		dataset.setValue("视频学 JAVA", 800);
		dataset.setValue("JAVA 全能速查宝典", 1000);
		return dataset;
	}
```

（2）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias()为 false，关闭抗锯齿。代码如下：

```
	public static JFreeChart getJFreeChart() {
		//获取数据集
		PieDataset dataset = getPieDataset();
		JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
		//关闭抗锯齿
		chart.setAntiAlias(false);
		return chart;
	}
```

（3）创建 setLegendTitle()方法，在该方法内获取 LegendTitle 对象，使用 LegendTitle 的 setPosition()方法把图例显示在图表右侧。

```
	public static void setLegendTitle(JFreeChart chart) {
		LegendTitle legendTitle = chart.getLegend();
		//设置图例位置
		legendTitle.setPosition(RectangleEdge.RIGHT);
	}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

心法领悟 177：JFreeChart 图例的位置。

对于 JFreeChart 图例的位置，默认采用的是 RectangleEdge.BOTTOM，即位于图表下方。在 RectangleEdge 中还有另外 3 个常量用于标识图例方位，除了本实例用到的 RectangleEdge.RIGHT，还有 RectangleEdge.LEFT 和 RectangleEdge.TOP。

第 8 章

基础图表技术

- 普通饼图
- 3D 饼图
- 多饼图
- 基本柱形图
- X 坐标轴
- Y 坐标轴
- 高级柱形图

8.1　普 通 饼 图

实例 178　分离饼图

光盘位置：光盘\MR\08\178

高级

实用指数：★★★★★

实例说明

制作图表时，某一个或几个类别可能非常重要，需要把这一部分内容突出显示。如本实例突出显示 8 月份销售最多的书和最少的书，就要把《Java 范例完全自学手册（1DVD）》和《JAVA 全能速查宝典》分离出来，运行结果如图 8.1 所示。

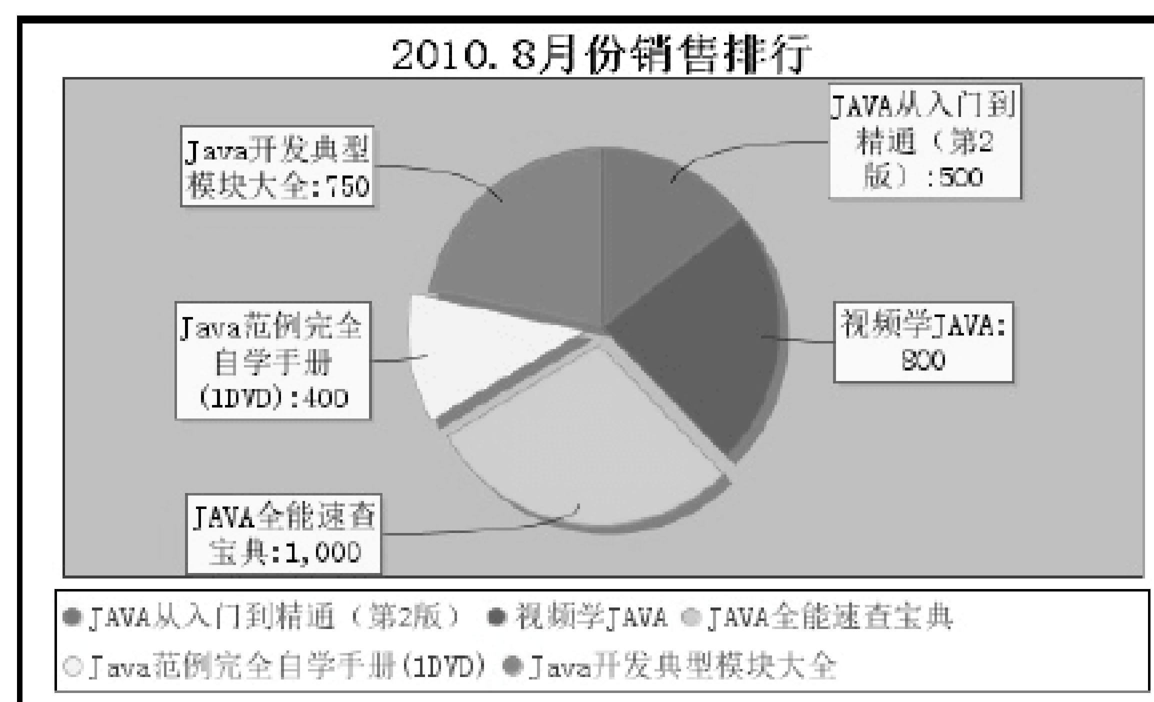


图 8.1　分离饼图

关键技术

使用 PiePlot 类的 setExplodePercent()方法可以指定要分离的饼图扇形。语法如下：

```
public void setExplodePercent(Comparable key, double percent)
```

参数说明

❶ key：表示要分离的名称。

❷ percent：表示要分离的距离。

使用方法如下：

```
//需要分离的图书
piePlot.setExplodePercent("Java 范例完全自学手册(1DVD)", 0.1);
piePlot.setExplodePercent("JAVA 全能速查宝典", 0.1);
```

（1）创建 ChartUtil 类，编写 getPieDataset()方法返回饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        DefaultPieDataset dataset = new DefaultPieDataset();
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        dataset.setValue("Java 范例完全自学手册(1DVD)", 400);
        dataset.setValue("Java 开发典型模块大全", 750);
        return dataset;
}
```

（2）编写 setPiePlotFont(JFreeChart chart)方法，设置饼图的字体。代码如下：

```
public static void setPiePoltFont(JFreeChart chart) {
        //图表（饼图）
        PiePlot piePlot = (PiePlot) chart.getPlot();
        piePlot.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        piePlot.setLabelGenerator(new StandardPieSectionLabelGenerator("{0}:{1}"));
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.BOLD, 20));
        //图例
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
}
```

（3）编写 createPiePlot(JFreeChart chart)方法，将饼图指定的分类区域分离。代码如下：

```
public static void createPiePlot(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
```

```
        //需要分离的图书
        piePlot.setExplodePercent("Java 范例完全自学手册(1DVD)", 0.1);
        piePlot.setExplodePercent("JAVA 全能速查宝典", 0.1);
}
```

（4）编写 getJFreeChart()方法，生成 JFreeChart 的分离饼图。代码如下：

```
public static JFreeChart getJFreeChart() {
        PieDataset dataset = getPieDataset();       //获取饼图数据集合
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        setPiePoltFont(chart);                      //设置字体
        createPiePlot(chart);                       //分离饼图
        return chart;
}
```

（5）创建 index.jsp 页面，显示 JFreeChart 生成的分离饼图。具体代码参见配书光盘。

秘笈心法

心法领悟 178：Plot 类。

在 JFreeChart 组件中，用 Plot 抽象类表示图表的绘图区对象。绘图区对象是 JFreeChart 组件中的一个重要对象，使用过程中可以通过此类设置绘图区属性及样式。Plot 类包含一些常用的派生类，如 PiePlot 类（饼图的绘图区对象）、CategoryPlot 类（支持折线图、区域图等的绘图区对象）和 XYPlot 类（表示具有 XY 轴数据的绘图区对象）。

实例 179	椭圆形饼图 光盘位置：光盘\MR\08\179	高级 实用指数：★★★★

实例说明

一般的饼图是圆形的，但有时也会使用椭圆形的饼图。利用 JFreeChart 提供的绘制椭圆饼图功能，可以非常方便、快捷地绘制一个椭圆形的饼图，如图 8.2 所示。

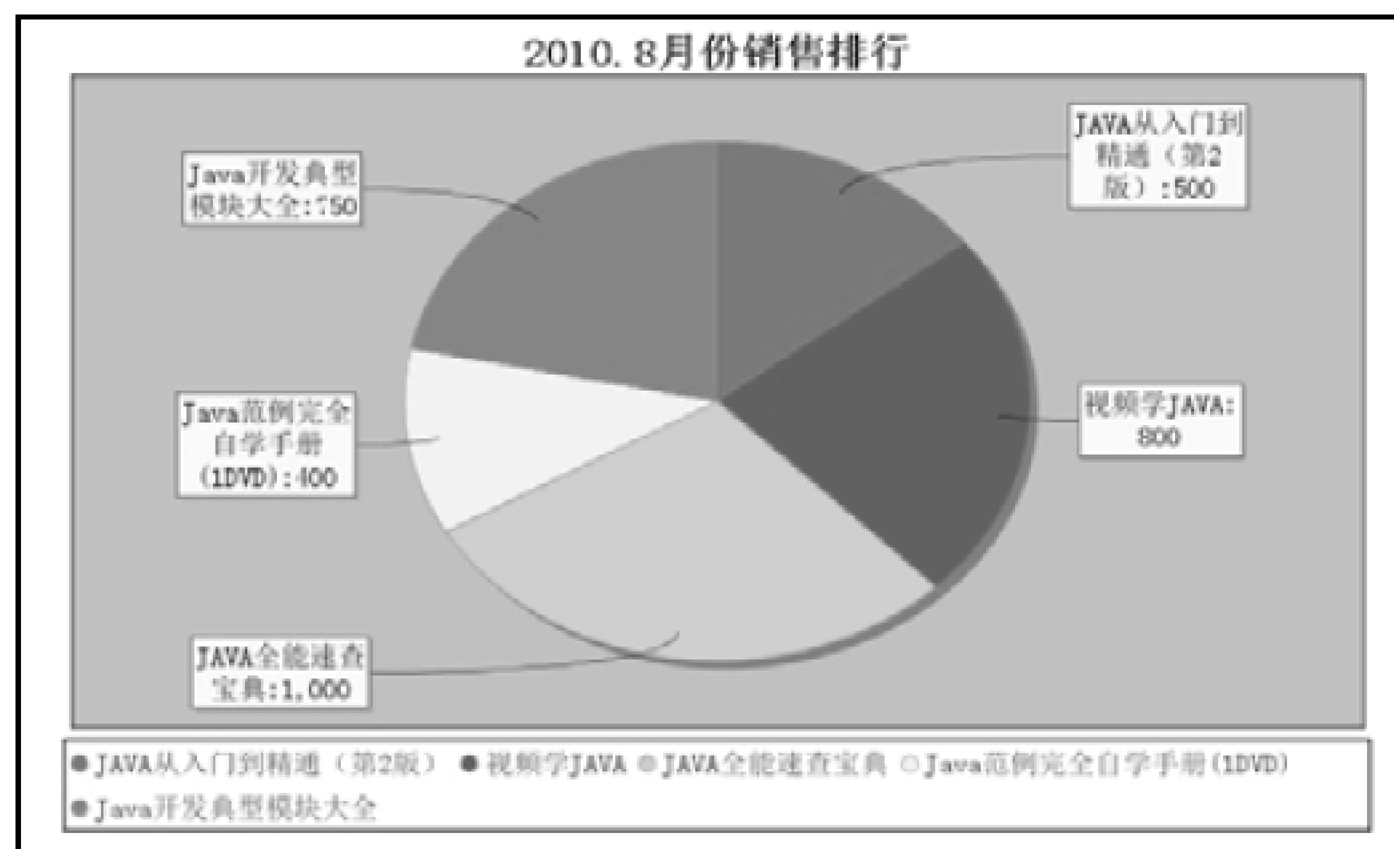


图 8.2　椭圆形饼图

关键技术

PiePlot 类提供的 setCircular()方法可以指定是否绘制圆形。其语法如下：

```
public void setCircular(boolean flag)
```

参数说明

flag：表示要绘制的图表是否为圆形。当 flag 为 true 时，表示要绘制的图表为正圆形；flag 为 false 时，表示要绘制的图表为椭圆形。

（1）创建 ChartUtil 类，编写 createPiePlot()方法，在该方法中调用 PiePlot 的 setCircular()方法，设置图表为

椭圆形。代码如下：

```
public static void createPiePlot(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
        PiePlot.setCircular(false);                    //是否为椭圆
}
```

（2）编写 getJFreeChart()方法，获取 DefaultPieDataset 创建的饼图的数据集合，然后使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，再修改饼图字体。代码如下：

```
public static JFreeChart getJFreeChart() {
        PieDataset dataset = getPieDataset();      //获取数据集
        //生成 JFreeChart 对象
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        setPiePoltFont(chart);                     //设置图表字体
        createPiePlot(chart);                      //设置饼图形状
        return chart;
}
```

（3）创建 index.jsp 页面，显示 JFreeChart 生成的椭圆形饼图。具体代码参见配书光盘。

秘笈心法

心法领悟 179：饼图的形状。

大多数情况下的饼图都是圆形，不过可以通过 setCircular()方法来设置饼图是圆形还是椭圆形；如果不使用该方法进行设置，JFreeChart 默认情况下都是圆形饼图。

实例 180	饼图的阴影 光盘位置：光盘\MR\08\180	高级 实用指数：★★★

实例说明

JFreeChart 饼图可以在生成时带有图表阴影，并且在图表的阴影区域中不同分类之间还绘制了分隔线，如本实例就实现了这样的效果，如图 8.3 所示。

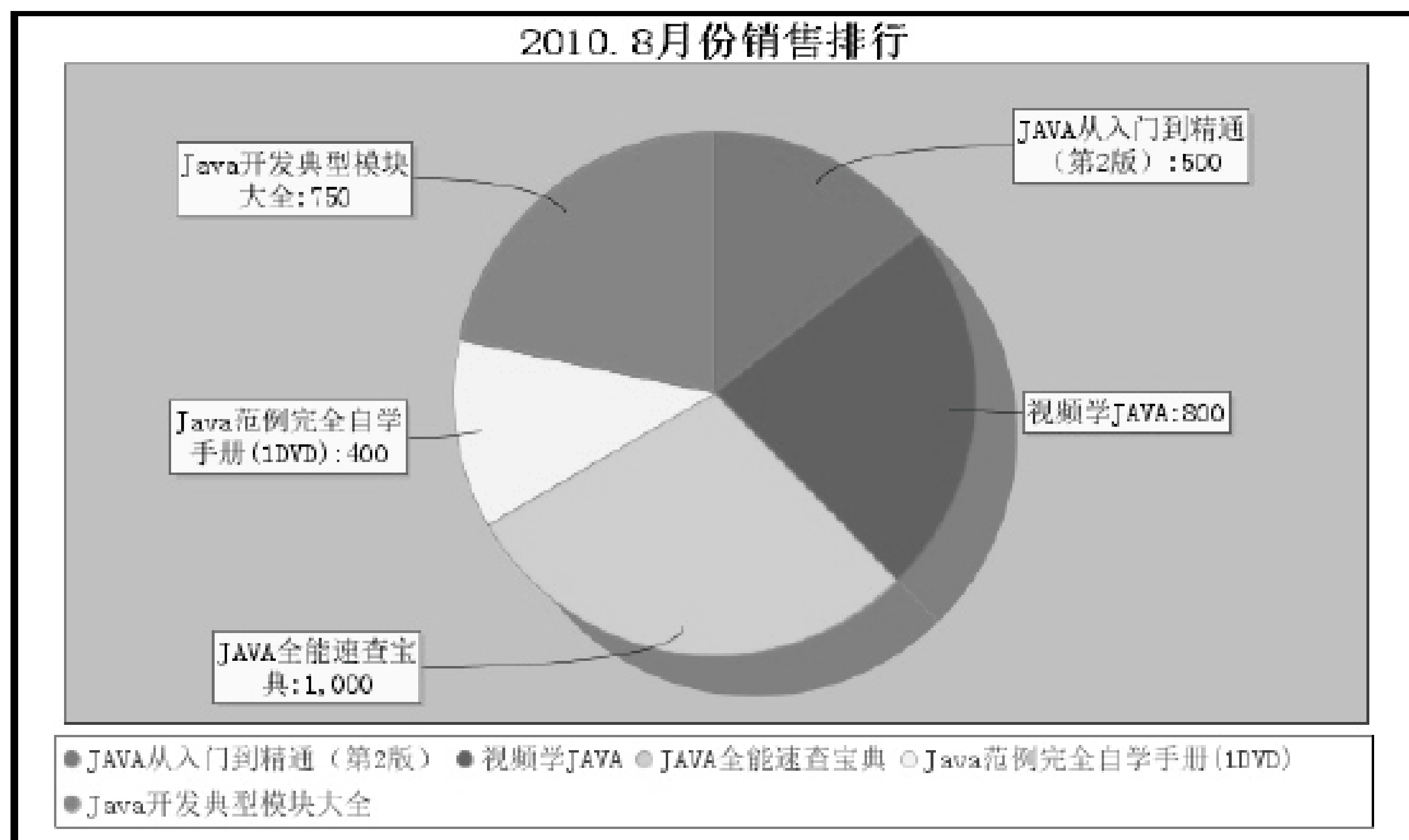


图 8.3　饼图的阴影

对于 JFreeChart 产生的图表阴影，可以使用如下两个方法设置偏移。

（1）使用 setShadowXOffset()方法设置 X 轴方向的偏移。语法如下：

```
public void setShadowXOffset(double offset)
```

参数说明

offset：表示 X 轴方向上的偏移量。

（2）使用 setShadowYOffset()方法设置 Y 轴方向的偏移。语法如下：

```
public void setShadowYOffset(double offset)
```

参数说明

offset：表示 Y 轴方向上的偏移量。

设计过程

（1）创建 ChartUtil 类，编写 createPiePlot()方法，该方法中使用 PiePlot 的 setShadowXOffset(double x)和 setShadowYOffset(double y)方法设置阴影坐标都为 20。代码如下：

```
public static void createPiePlot(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置阴影效果
        piePlot.setShadowXOffset(20);
        piePlot.setShadowYOffset(20);
}
```

（2）编写 getJFreeChart()方法，获取 DefaultPieDataset 创建的饼图的数据集合，然后使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，再修改饼图字体。代码如下：

```
public static JFreeChart getJFreeChart() {
        PieDataset dataset = getPieDataset();                                                    //获取数据集
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);    //生成 JFreeChart 对象
        setPiePoltFont(chart);                                                                   //设置图表字体
        createPiePlot(chart);                                                                    //设置饼图的阴影
        return chart;
}
```

（3）创建 index.jsp 页面，显示 JFreeChart 生成的饼图。具体代码参见配书光盘。

秘笈心法

心法领悟 180：设置饼图的阴影。

饼图阴影的 X、Y 轴坐标默认设置都为 0，使用时可以根据需要设置阴影的显示坐标。如果不希望使用阴影效果，可以设置阴影的 X、Y 轴的坐标均为 0。

实例 181 加粗饼图分类边框

光盘位置：光盘\MR\08\181

高级

实用指数：★★★

实例说明

饼图分类的边框可能会根据不同需求进行改变，本实例将所有分类的边框加粗，通过修改边框笔触达到预想的效果，如图 8.4 所示。

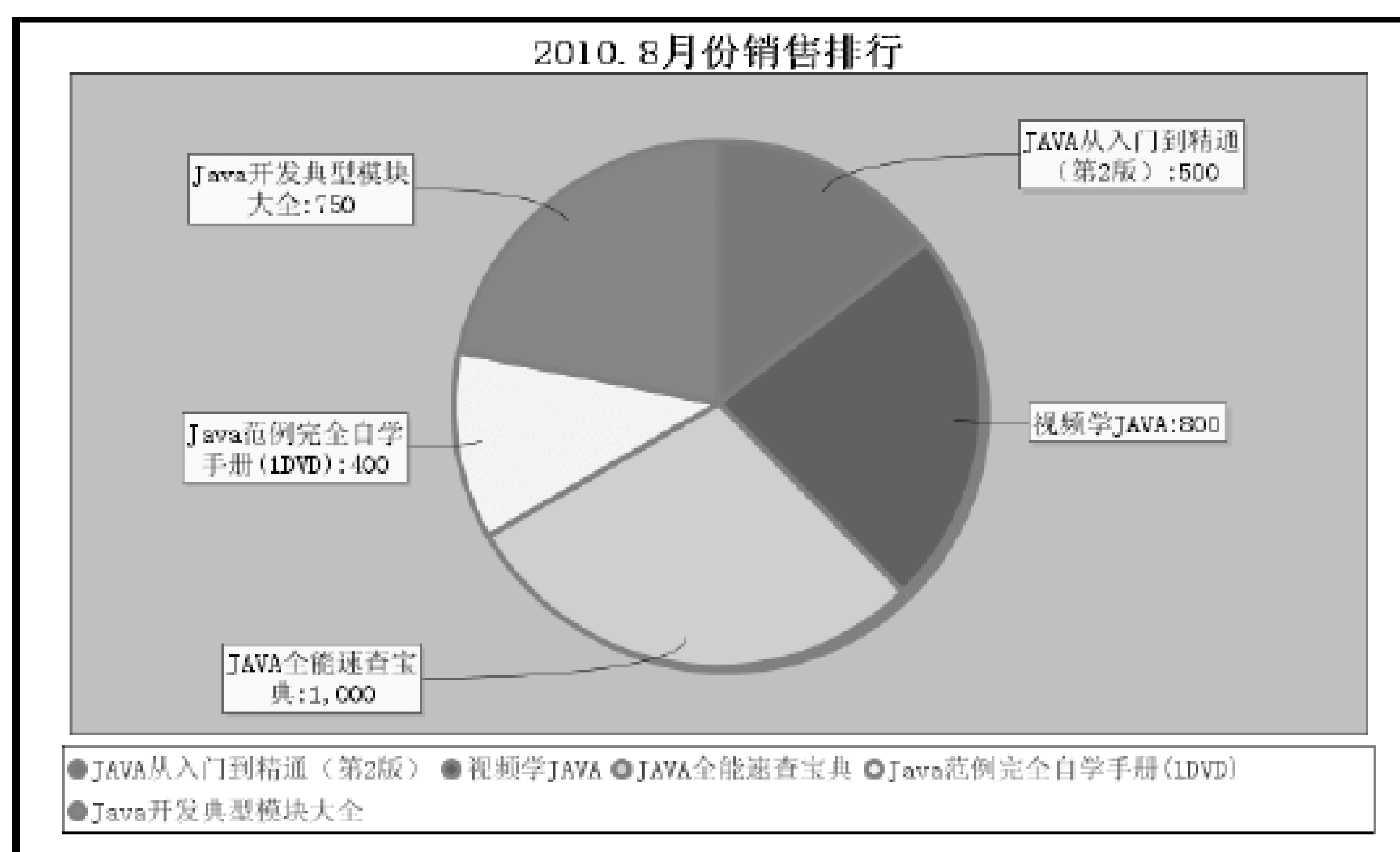


图 8.4 加粗分类边框

关键技术

PiePlot 类中的 setSectionOutlineStroke()方法主要用于设置饼图的笔触。语法如下：

```
public void setSectionOutlineStroke(Comparable key, Stroke stroke)
```

参数说明

❶ key：表示要设置笔触的分类名称。

❷ stroke：表示要设置的笔触对象。

设计过程

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
	//创建数据集合实例
	DefaultPieDataset dataset = new DefaultPieDataset();
	//向数据集合添加数据
	dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
	dataset.setValue("视频学 JAVA", 800);
	dataset.setValue("JAVA 全能速查宝典", 1000);
	dataset.setValue("Java 范例完全自学手册(1DVD)", 400);
	dataset.setValue("Java 开发典型模块大全", 750);
	return dataset;
}
```

（2）创建 createPiePlot()方法，在该方法中调用 PiePlot 的 setSectionOutlineStroke()方法为所有分类设置边框笔触为 3。代码如下：

```
public static void createPiePlot(JFreeChart chart) {
	PiePlot piePlot = (PiePlot) chart.getPlot();
	//设置饼图边框笔触
	piePlot.setSectionOutlineStroke("JAVA 从入门到精通（第 2 版）",new BasicStroke(3f));
	piePlot.setSectionOutlineStroke("视频学 JAVA", new BasicStroke(3f));
	piePlot.setSectionOutlineStroke("JAVA 全能速查宝典", new BasicStroke(3f));
	piePlot.setSectionOutlineStroke("Java 范例完全自学手册(1DVD)", new BasicStroke(3f));
	piePlot.setSectionOutlineStroke("Java 开发典型模块大全", new BasicStroke(3f));
}
```

（3）使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，并且设置饼图字体，加粗饼图的分类边框。代码如下：

```
public static JFreeChart getJFreeChart() {
	//获取数据集
	PieDataset dataset = getPieDataset();
	JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
	setPiePoltFont(chart);
	createPiePlot(chart);
	return chart;
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 181：分类的内容和分类的图示。

分类的内容和分类的图示是关联在一起的，如在本实例中设置分类的边框笔触为 3，那么分类图示的边框笔触也会随之变为 3。

实例 182	设置饼图颜色 光盘位置：光盘\MR\08\182	高级 实用指数：★★★★

饼图分类的颜色一般都是通过 JFreeChart 自动设置，也可以由开发人员指定。本实例将对饼图分类的颜色进行设置，运行结果如图 8.5 所示。

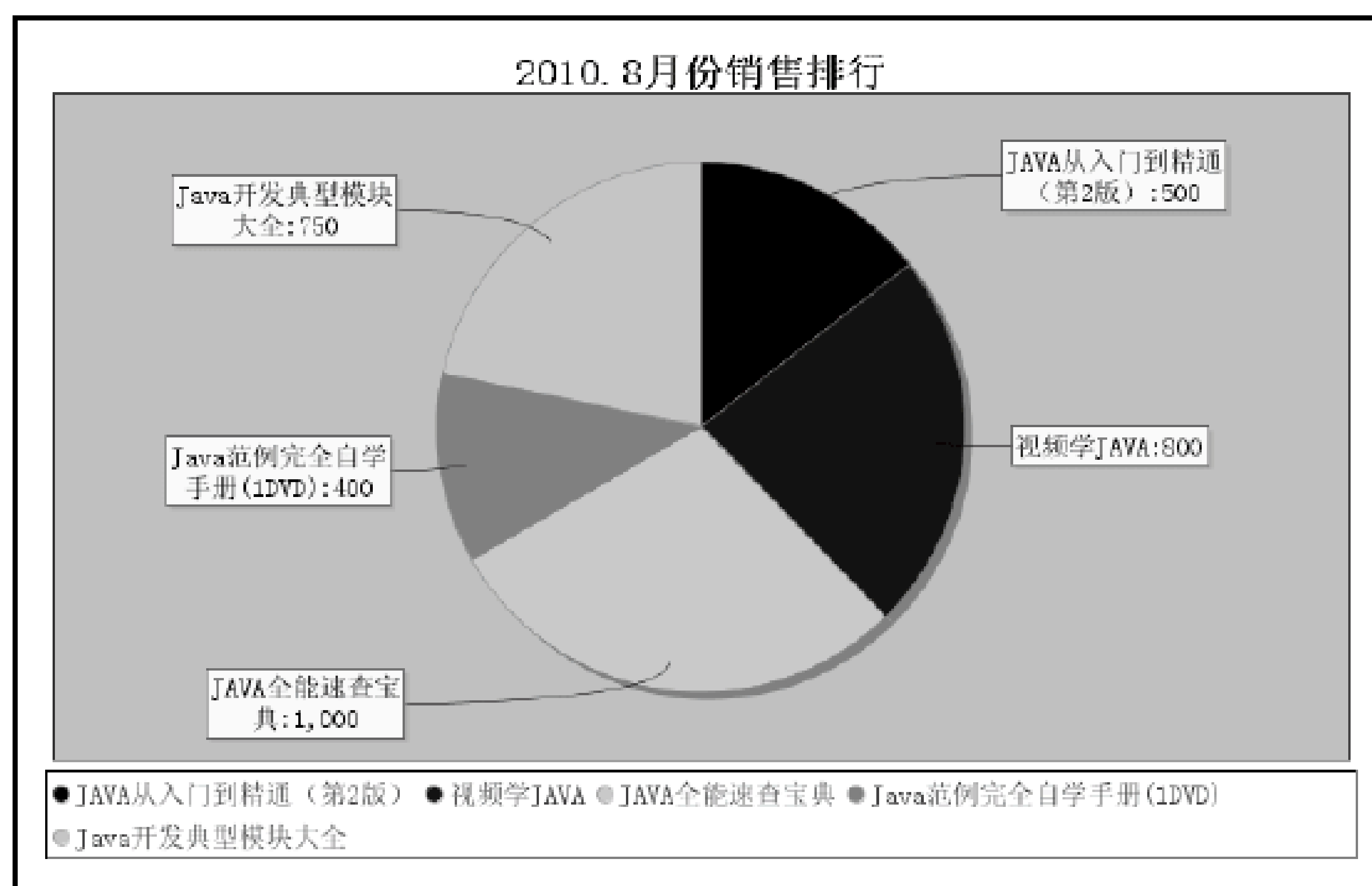


图 8.5　设置饼图颜色

关键技术

PiePlot 类中的 setSectionPaint()方法主要用于设置饼图分类的颜色。语法如下：

```
public void setSectionPaint(Comparable key, Paint paint)
```

参数说明

❶ key：表示要设置颜色的分类名称。

❷ paint：表示要设置的颜色。

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        //创建数据集合实例
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向数据集合添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        dataset.setValue("Java 范例完全自学手册(1DVD)", 400);
        dataset.setValue("Java 开发典型模块大全", 750);
        return dataset;
}
```

（2）创建 createPiePlot()方法，在该方法中调用 PiePlot 类的 setSectionPaint()方法为所有分类设置颜色。代码如下：

```
public static void createPiePlot(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置饼图的颜色
        piePlot.setSectionPaint("JAVA 从入门到精通（第 2 版）", Color.black);
        piePlot.setSectionPaint("视频学 JAVA", Color.blue);
        piePlot.setSectionPaint("JAVA 全能速查宝典", Color.cyan);
        piePlot.setSectionPaint("Java 范例完全自学手册(1DVD)", Color.gray);
        piePlot.setSectionPaint("Java 开发典型模块大全", Color.orange);
}
```

（3）使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，并且设置饼图字体和颜色。代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        setPiePoltFont(chart);
        createPiePlot(chart);
        return chart;
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 182：为饼图设置分类颜色。

为饼图设置分类颜色时，可以为一部分分类设置颜色，也可为所有分类设置颜色。如果只为其中一部分设置颜色，那么 JFreeChart 会自动设置剩余分类的颜色。

实例 183　饼图旋转角度和顺序

光盘位置：光盘\MR\08\183

高级

实用指数：★★★

实例说明

通常情况下，饼图显示为一个圆形的区域，每个类别都占据其中一块扇形区域，其位置可以因饼图的旋转而改变。饼图类别的排列是按照数据集添加的顺序显示的。默认情况下，饼图是按照顺时针的顺序根据数据集添加的先后顺序依次向下排列。本实例修改了默认顺序，使饼图能够逆时针进行排列，运行结果如图 8.6 所示。

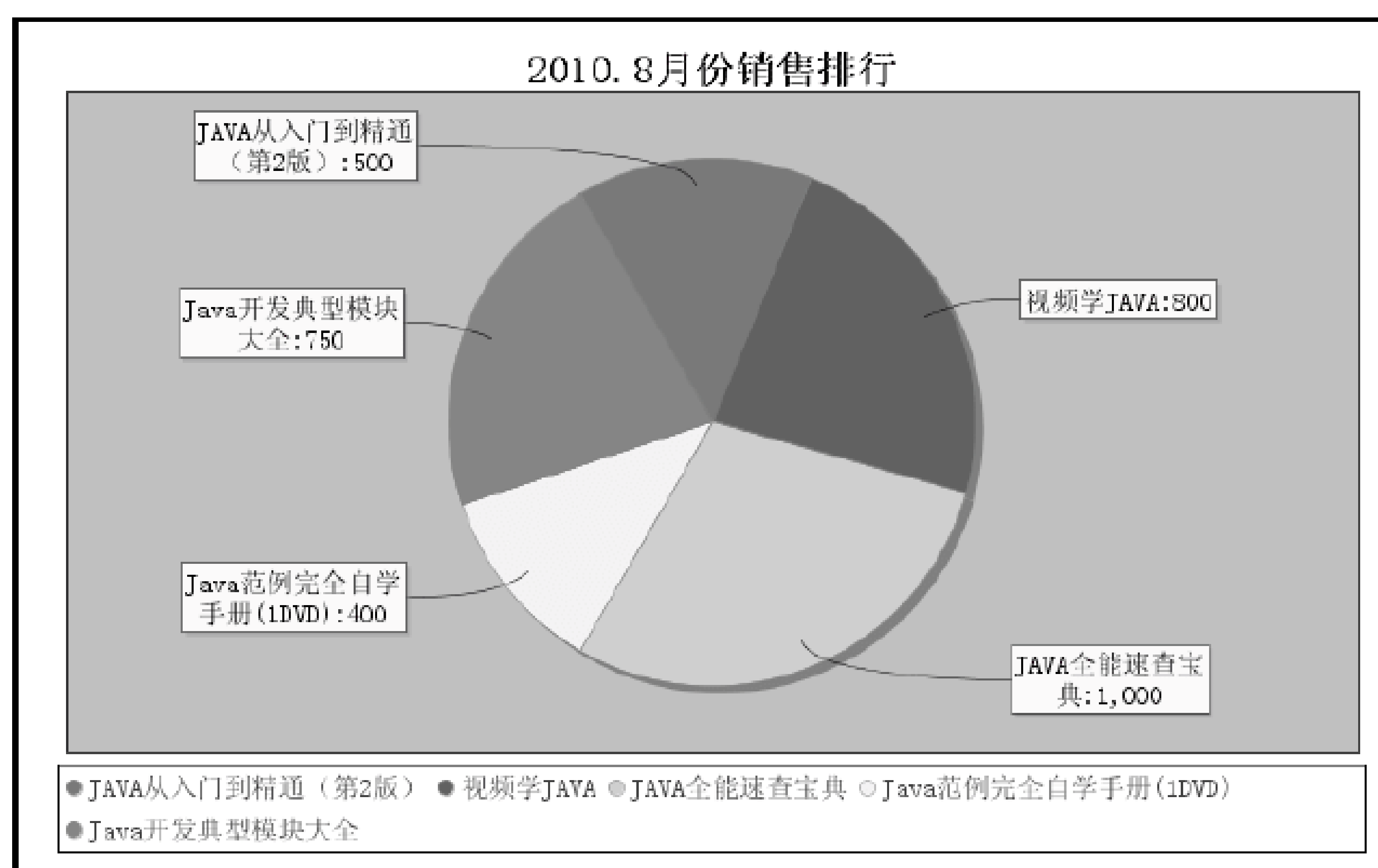


图 8.6　饼图旋转角度和顺序

关键技术

（1）饼图默认情况下初始角度为 90°，使用 PiePlot 类的 setStartAngle()方法可以设置旋转角度。语法如下：

```
public void setStartAngle(double angle)
```

参数说明

angle：表示要改变的饼图初始角度。

（2）使用 PiePlot 类的 setDirection()方法可以设置饼图的排列顺序。语法如下：

```
public void setDirection(Rotation direction)
```

参数说明

direction：表示饼图的排列顺序。direction 是 Rotation 类的实例，在 Rotation 中已经定义了如下两个常量用于设置饼图分类的排列顺序。

- Rotation. CLOCKWISE：定义了饼图顺时针排列分类。
- Rotation.ANTICLOCKWISE：定义了饼图逆时针排列分类。

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
	//创建数据集合实例
	DefaultPieDataset dataset = new DefaultPieDataset();
```

```
        //向数据集合添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        dataset.setValue("Java 范例完全自学手册(1DVD)", 400);
        dataset.setValue("Java 开发典型模块大全", 750);
        return dataset;
}
```

（2）创建 createPiePlot()方法，在该方法中调用 PiePlot 类的 setStartAngle()方法，设置初始角度为 120°，然后设置为逆时针旋转。代码如下：

```
public static void createPiePlot(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
        piePlot.setStartAngle(120);                          //设置旋转角度
        piePlot.setDirection(Rotation.ANTICLOCKWISE);      //设置逆时针

}
```

（3）使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，并且设置饼图字体和旋转角度。代码如下：

```
public static JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        setPiePoltFont(chart);
        createPiePlot(chart);
        return chart;
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 183：JFreeChart 图表的角度。

JFreeChart 产生的图表角度是以数据集中添加的第一个分类的扇形左边线为基准线，以饼图中心为圆点，以窗体的下边线作为参照物。本实例中的角度是指基准线与参照物逆时针所成的角度，运行效果如图 8.7 所示。

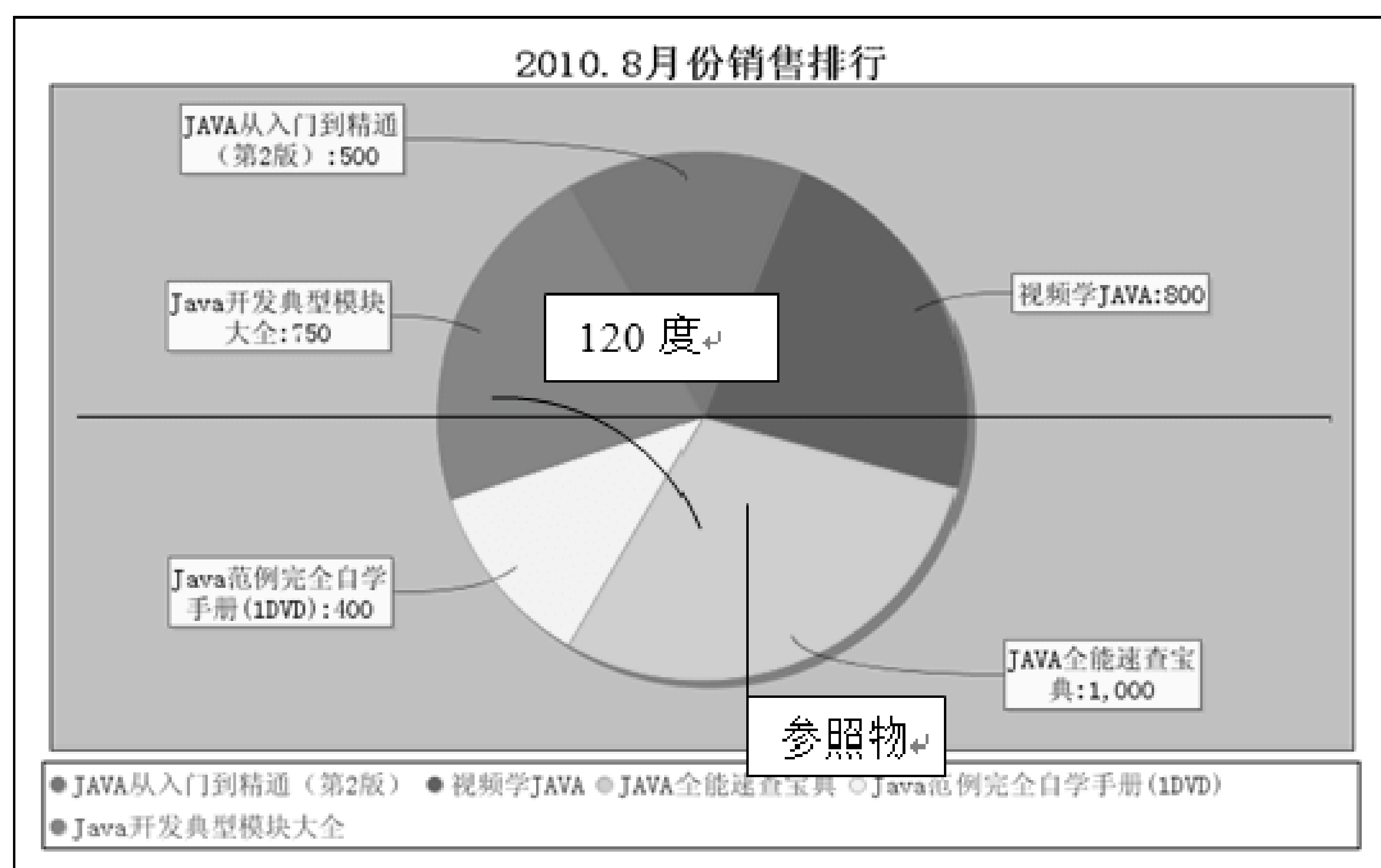


图 8.7　图表的角度

实例 184　隐藏分类标签连接线

光盘位置：光盘\MR\08\184

高级

实用指数：★★★

饼图的分类标签默认都是使用连接线连接的，如果把连接线去掉，那么饼图的分类标签将直接显示在分类图上，如图 8.8 所示。

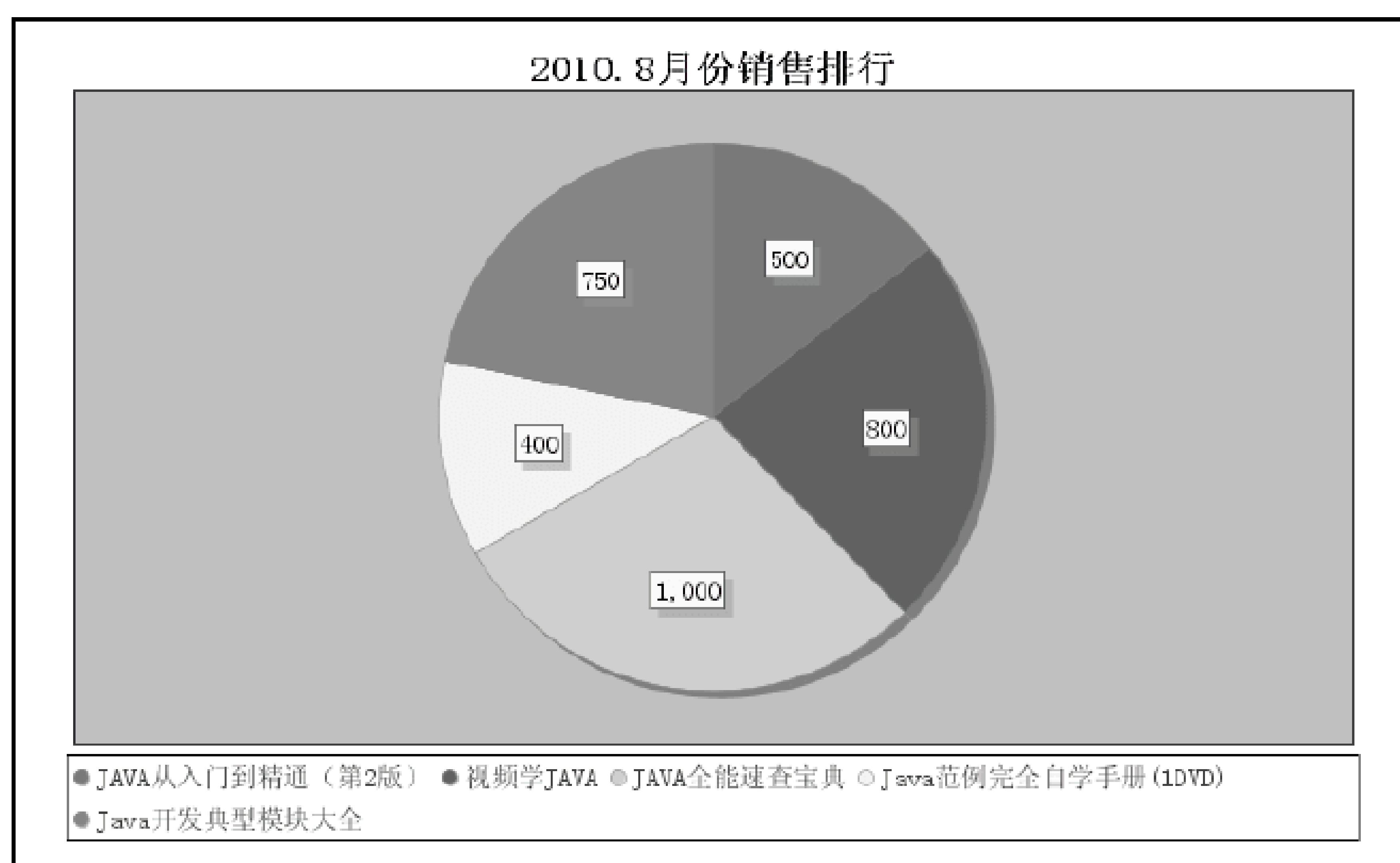


图 8.8　隐藏分类标签连接线

关键技术

饼图的分类标签连接线可以使用 PiePlot 类的 setSimpleLabels(boolean simple)方法进行设置。语法如下：

```
public void setSimpleLabels(boolean simple)
```

参数说明

simple：当 simple 为 true 时，表示使用简单的分类标签，连接线不会显示；当 simple 为 false 时，表示不使用简单的分类标签，连接线将显示出来。

设计过程

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        //创建数据集合实例
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向数据集合中添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        dataset.setValue("Java 范例完全自学手册(1DVD)", 400);
        dataset.setValue("Java 开发典型模块大全", 750);
        return dataset;
}
```

（2）创建 createPiePlot()方法，在该方法中设置 PiePlot 类的 setSimpleLabels()方法为 true。代码如下：

```
public static void createPiePlot(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
        piePlot.setSimpleLabels(true);              //设置标签模式
}
```

（3）使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        setPiePoltFont(chart);                      //设置饼图使用的字体
        createPiePlot(JFreeChart chart);            //隐藏分类标签连接线
        return chart;
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 184：分类标签。

如果分类标签的内容比较短，则可以像本实例这样，使用 setSimpleLabels()方法隐藏分类标签连接线，将分

类标签直接显示在分类图上；如果分类标签名称很长，一个标签内容可能会占用多个分类图，这时就不容易区分标签真正代表的分类。

8.2　3D 饼图

实例 185　创建 3D 饼图

光盘位置：光盘\MR\08\185

高级

实用指数：★★★★★

实例说明

普通的饼图都是平面的，可能会给人一种死板的感觉。利用 JFreeChart 提供的相应功能，可以很方便地创建 3D 饼图，使图表更活泼、直观。本实例将实现 3D 饼图的创建，运行结果如图 8.9 所示。

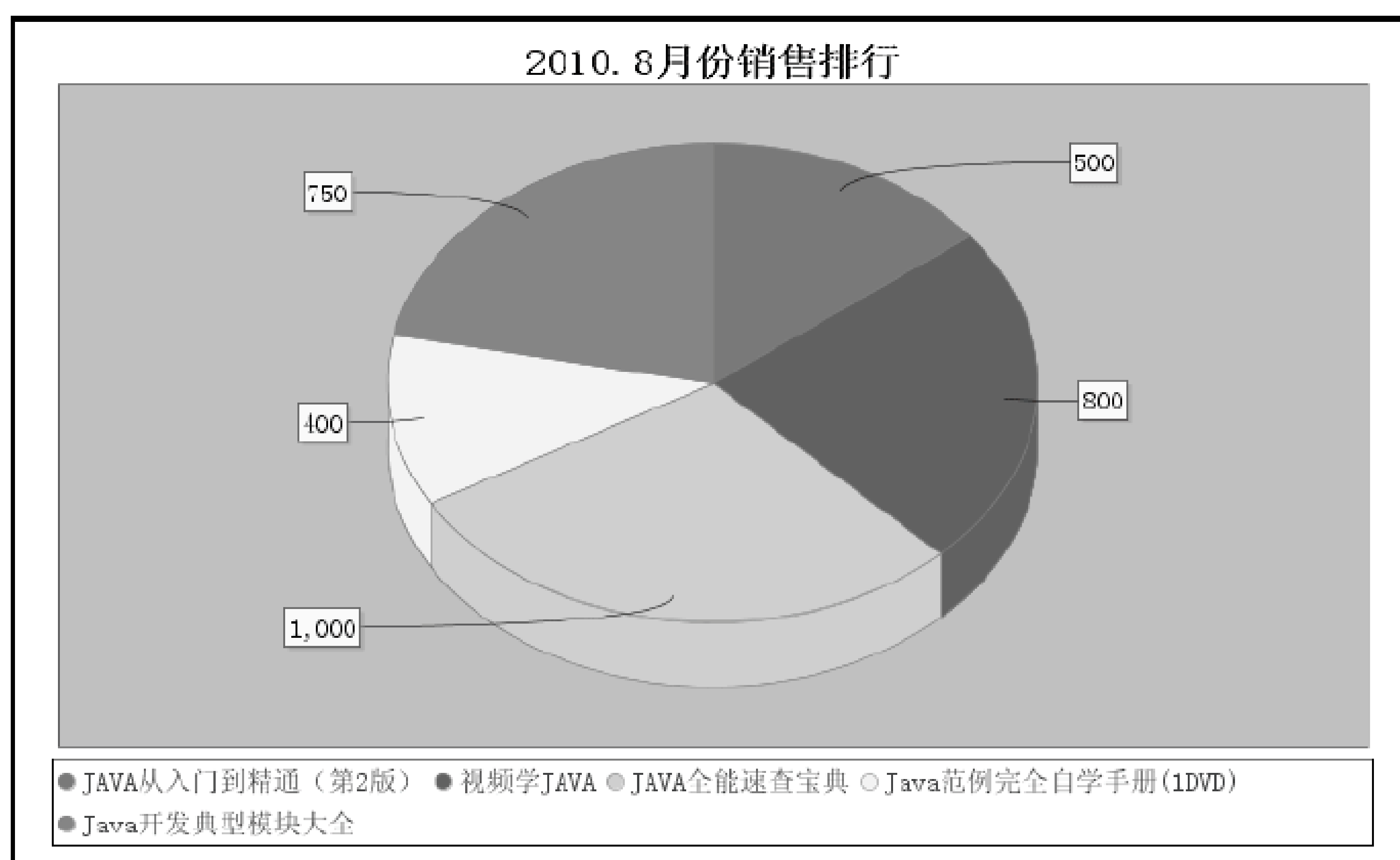


图 8.9　3D 饼图

关键技术

使用 ChartFactory 类的 createPieChart3D()方法可以创建 3D 饼图。语法如下：

```
public static JFreeChart createPieChart3D(String title, PieDataset dataset, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：表示饼图的标题。

❷ dataset：表示饼图的数据集合。

❸ legend：表示是否使用图示。

❹ tooltips：表示是否生成工具栏提示。

❺ urls：表示是否生成 URL 链接。

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        //创建数据集合实例
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向数据集合中添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        dataset.setValue("Java 范例完全自学手册(1DVD)", 400);
```

```
        dataset.setValue("Java 开发典型模块大全", 750);
        return dataset;
}
```

（2）使用 ChartFactory 类根据饼图的数据集合创建有 3D 效果的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart3D("2010.8 月份销售排行", dataset,true, true, false);
        setPiePoltFont(chart); //设置饼图使用的字体
        return chart;
}
```

（3）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 185：3D 饼图的效果。

3D 饼图和普通的饼图有些效果是通用的，如分离饼图、旋转饼图初始角度、设置简单的分类标签等，但是也有一部分效果是其特有的，如 3D 饼图不能设置阴影效果。



实例 186　**3D 饼图透明度**　高级

光盘位置：光盘\MR\08\186　实用指数：★★★★

实例说明

3D 饼图与普通饼图相比立体感很强，本实例通过给 3D 饼图增加透明度可以让饼图更具有立体感，运行效果如图 8.10 所示。

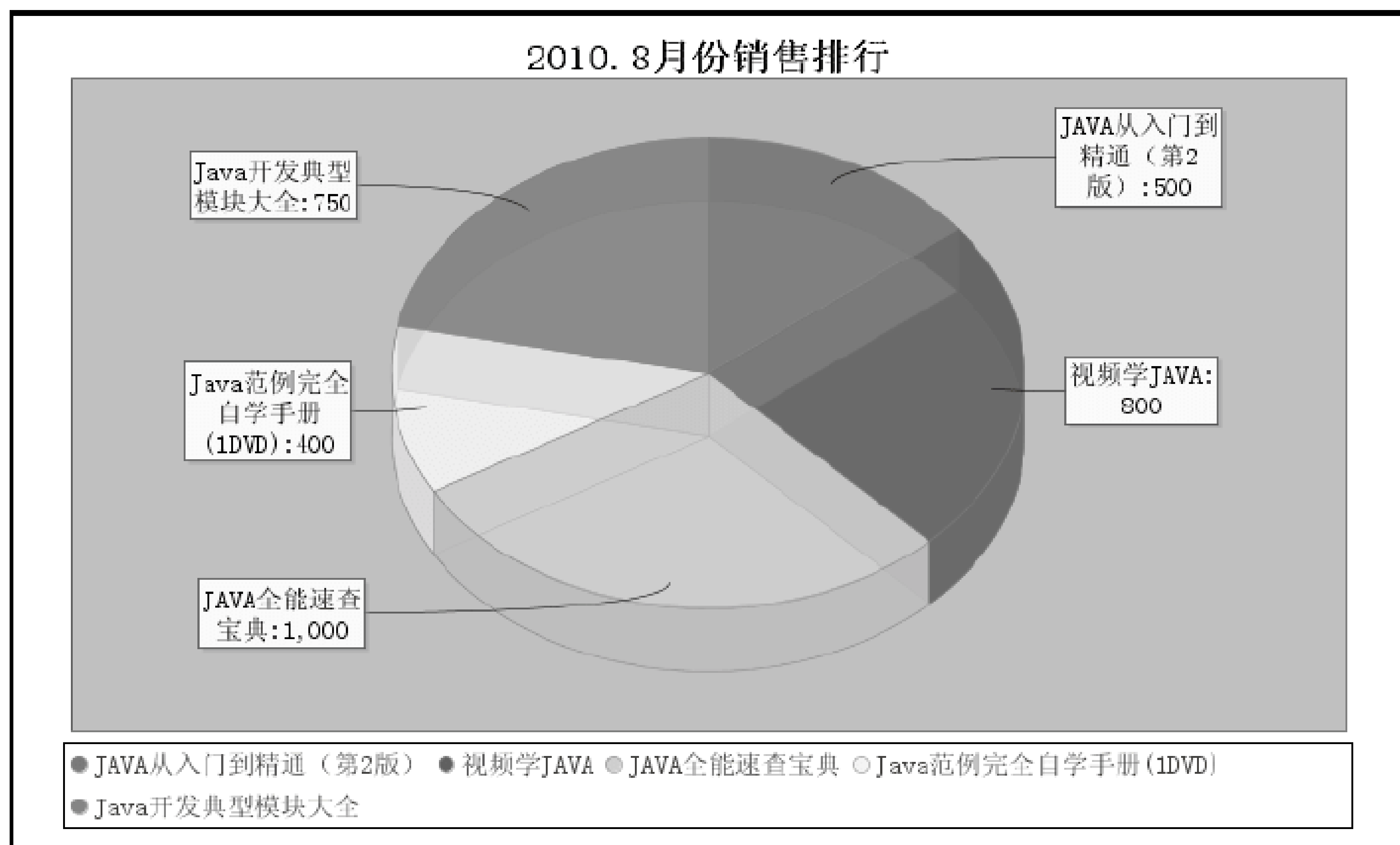


图 8.10　3D 饼图透明度

关键技术

要设置饼图的透明度，可以使用 PiePlot 类的 setForegroundAlpha()方法。语法如下：

```
public void setForegroundAlpha(float alpha)
```

参数说明

alpha：表示 3D 饼图的透明度，范围为 0～1，值越小，透明度越高，值越大，透明度越低。

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
        DefaultPieDataset dataset = new DefaultPieDataset();                    //创建数据集合实例
```

```
        //向数据集合中添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        dataset.setValue("Java 范例完全自学手册(1DVD)", 400);
        dataset.setValue("Java 开发典型模块大全", 750);
        return dataset;
}
```

（2）创建 createPiePlot()方法，设置透明度。代码如下：

```
public static void createPiePlot(JFreeChart chart) {
        PiePlot plot = (PiePlot)chart.getPlot();
        plot.setForegroundAlpha(0.7f);                                    //设置饼图透明度
}
```

（3）使用 ChartFactory 类根据饼图的数据集合创建有 3D 效果的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart3D("2010.8 月份销售排行",dataset, true, true, false);
        setPiePoltFont(chart);                                            //设置饼图使用的字体
        createPiePlot(chart);                                             //设置饼图的透明度
        return chart;
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 186：图表的透明度。

PiePlot 类的 setForegroundAlpha()方法用于设置图表的透明度，这个方法不只适用于 3D 的图表，还可以用在普通的图表中，不过为普通的图表使用透明度的效果没有 3D 图表的效果更突出。

实例 187　3D 饼图的 Z 轴

光盘位置：光盘\MR\08\187

高级　实用指数：★★★

3D 图表要比普通图表多一个 Z 轴，Z 轴一般是指 3D 图表的高度。默认情况下，JFreeChart 的 3D 饼图高度为 1.2。本实体将演示如何修改 3D 饼图的 Z 轴高度，运行结果如图 8.11 所示。

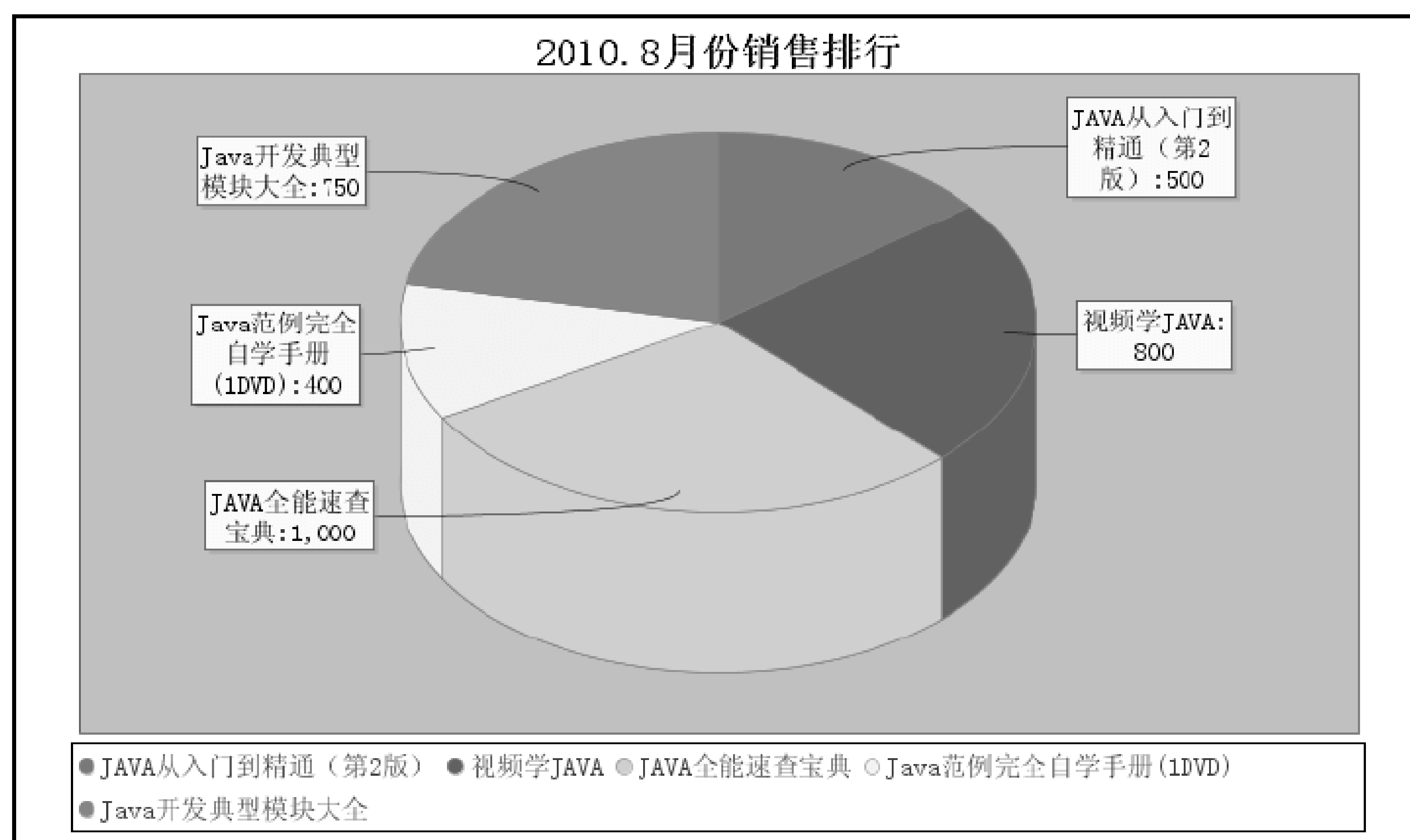


图 8.11　调整 3D 饼图的高度

关键技术

对于饼图 Z 轴的高度，可以使用 PiePlot3D 类的 setDepthFactor()方法进行设置。语法如下：

```
public void setDepthFactor(double factor)
```

参数说明

factor：表示 Z 轴的范围为 0～1，值越小，Z 轴越低，值越大，Z 轴越高。

设计过程

（1）创建 ChartUtil 类，使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private static PieDataset getPieDataset() {
	//创建数据集合实例
	DefaultPieDataset dataset = new DefaultPieDataset();
	//向数据集合中添加数据
	dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
	dataset.setValue("视频学 JAVA", 800);
	dataset.setValue("JAVA 全能速查宝典", 1000);
	dataset.setValue("Java 范例完全自学手册(1DVD)", 400);
	dataset.setValue("Java 开发典型模块大全", 750);
	return dataset;
}
```

（2）创建 createPiePlot()方法，在该方法中获取 3D 饼图的对象，然后设置 Z 轴的高度为 0.3。代码如下：

```
public static void createPiePlot(JFreeChart chart) {
	PiePlot3D plot    = (PiePlot3D)chart.getPlot();
	plot.setDepthFactor(0.3f);                          //设置 3D 饼图 Z 轴的高度
}
```

（3）使用 ChartFactory 类根据饼图的数据集合创建有 3D 效果的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
	PieDataset dataset = getPieDataset();
	JFreeChart chart = ChartFactory.createPieChart3D("2010.8 月份销售排行",dataset, true, true, false);
	setPiePoltFont(chart);                              //设置饼图使用的字体
	createPiePlot(JFreeChart chart);                    //设置 Z 轴的高度
	return chart;
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 187：设置 3D 饼图的侧面颜色。

一般情况下，3D 饼图的侧面颜色与上面的颜色一致，但是使用 setDarkerSides(boolean darker)方法可以让侧面颜色比上面的颜色更深，立体效果更强。当 darker 为 false（默认设置）时，表示不使用特殊的立体效果；当 darker 为 true 时，则使用这种特殊效果。

8.3　多　饼　图

实例 188	实现多饼图 光盘位置：光盘\MR\08\188	高级 实用指数：★★★★★



利用 JFreeChart 提供的图表功能可以建立多饼图。这里所说的多饼图并不是创建多个简单的饼图，而是创建一个饼图组。本实例中将为部门 1 和部门 2 创建 4—6 月份的销售统计图表，运行结果如图 8.12 所示。

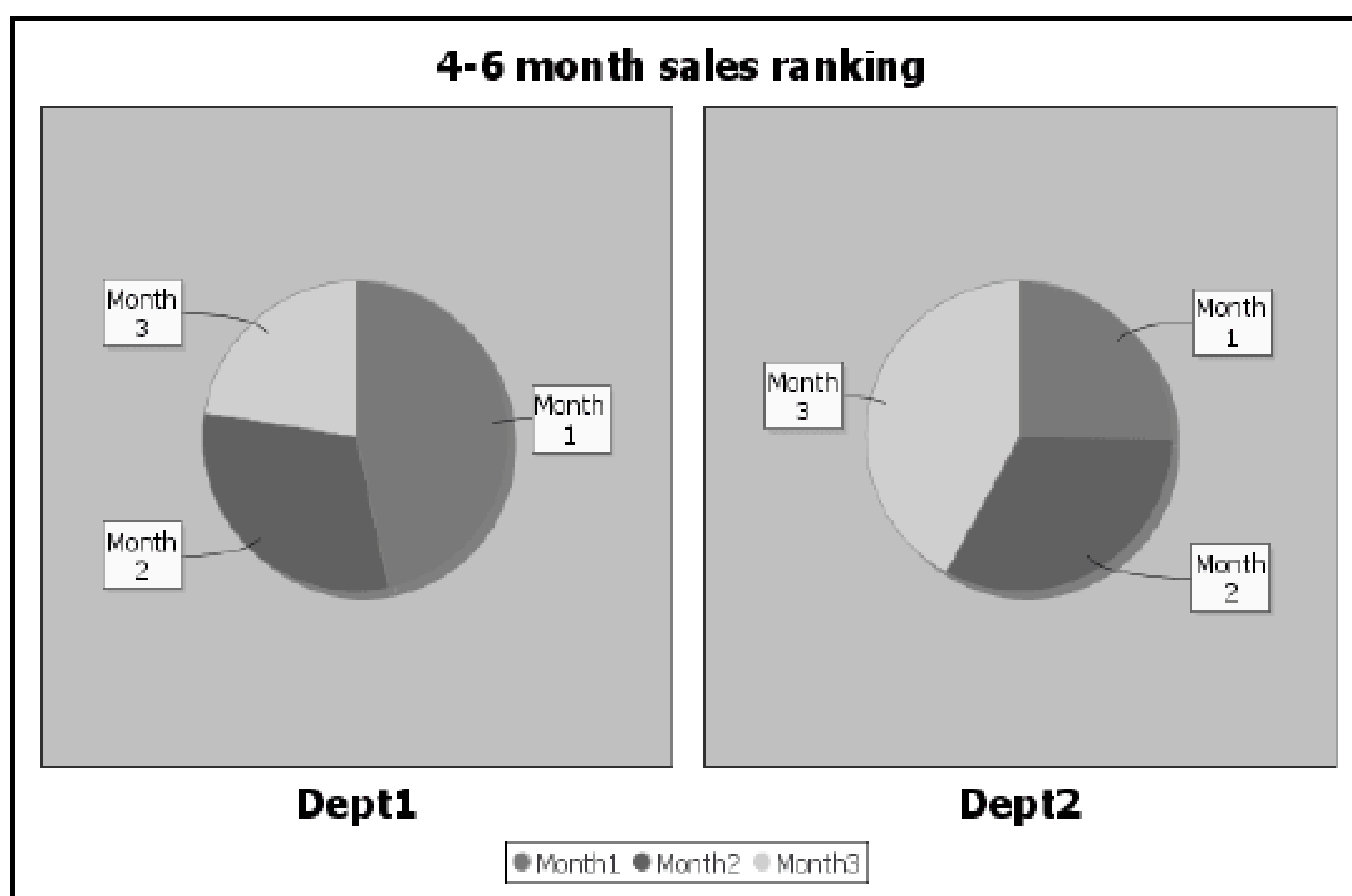


图 8.12　多饼图

关键技术

DatasetUtilities 类中包含有创建分类数据集的方法——createCategoryDataset()，创建完成后返回 CategoryDataset。语法如下：

```
public static CategoryDataset createCategoryDataset(String rowKeyPrefix, String columnKeyPrefix, double[][] data)
```

参数说明

❶ rowKeyPrefix：表示分类数据集的行名称。

❷ columnKeyPrefix：表示分类数据集的列名称。

❸ data：一个二维数组，根据行、列的名称存储数据。

ChartFactory 类中有很多方法，用于创建不同的 JFreeChart。例如，利用以下方法可以创建一个多饼图的 JFreeChart 对象。

```
createMultiplePieChart(String title,CategoryDataset dataset, TableOrder order, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：表示多饼图的窗体的标题。

❷ dataset：表示多饼图的数据集合。

❸ order：设置多饼图的显示方式。

❹ legend：表示是否使用图示。

❺ tooltips：表示是否生成工具栏提示。

❻ urls：表示是否生成 URL 链接。

设计过程

（1）创建 ChartUtil 类，编写 createDataset()方法，在该方法中设置图表数据，创建一个 CategoryDataset 对象。代码如下：

```
private static CategoryDataset createDataset() {
        double[ ][ ] data = new double[ ][ ] {
                        { 620, 410, 300 },
                        { 300, 390, 500 } };
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset(
                        "Dept",                         //行名称
                        "Month",                        //列名称
                        data);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中根据数据集生成多饼图的 JFreeChart 对象，并且指定多饼图排序方式为行排列。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = createDataset();      //获取数据集
        //生成 JFreeChart 对象
```

```
        JFreeChart chart = ChartFactory.createMultiplePieChart(
                        "4-6 month sales ranking ",          //饼图标题
                        dataset,                             //数据集
                        TableOrder.BY_ROW,                   //排序方式
                        true, true, false);
        return chart;
}
```

（3）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 188：多饼图的数据集。

在多饼图中创建数据集时需要指定数据集的行、列名称，生成多饼图以后 JFreeChart 会根据该名称自动为饼图分类，同时各个饼图将自动生成自己的名称（以指定的行、列名称为基础，后面添加相应数字）。

实例 189　多饼图乱码

光盘位置：光盘\MR\08\189

高级　实用指数：★★★

在 JFreeChart 的多饼图中，有时会出现中文乱码问题，这是因为字体造成的。普通的饼图乱码解决方案只能解决多饼图窗体和图示的乱码，其内部还是有乱码出现的，如图 8.13 所示。

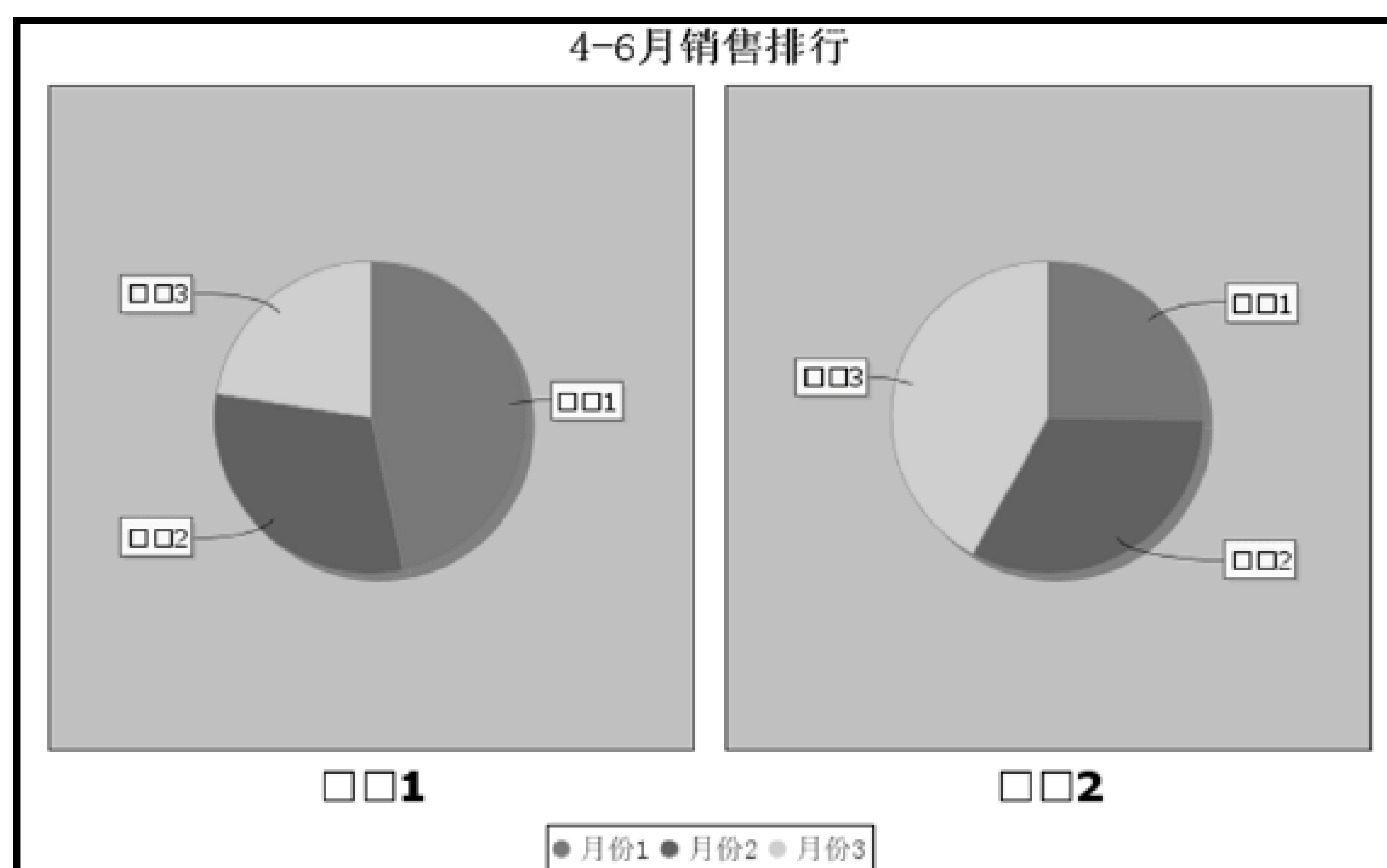


图 8.13　多饼图的乱码

这说明以前的饼图乱码解决方案不适用于多饼图。本实例将解决多饼图中饼图内部的乱码问题，运行结果如图 8.14 所示。

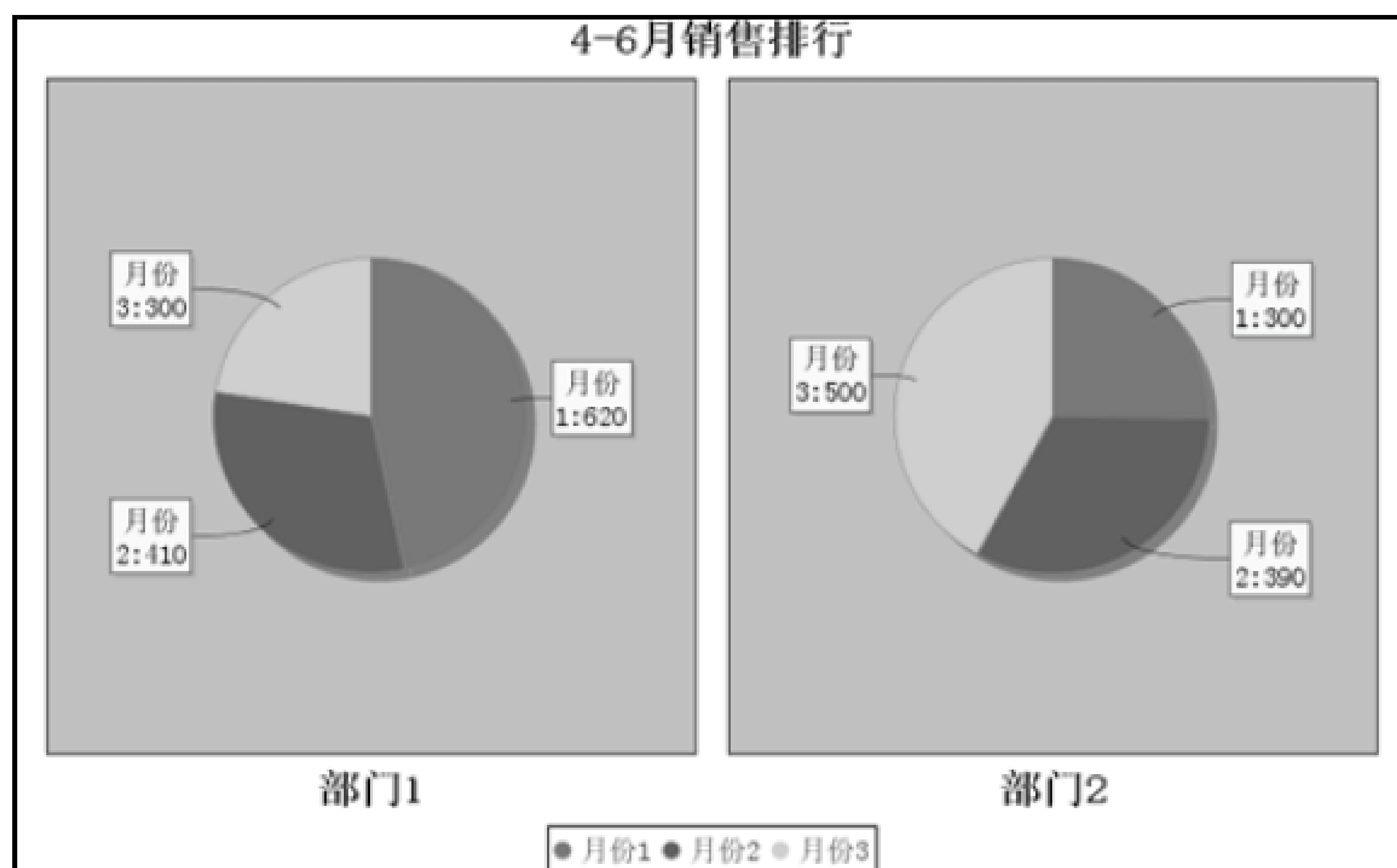


图 8.14　多饼图

关键技术

普通饼图使用 JFreeChart 的 getPlot()方法得到的实例是 PiePlot 的对象，而多饼图使用 JFreeChart 的 getPlot()方法得到的实例是 MultiplePiePlot 的对象。如果希望得到多饼图的饼图对象，要使用 MultiplePiePlot 的 getPieChart()方法获取 JFreeChart 对象，然后再从 JFreeChart 的对象里获取 PiePlot 的实例。代码如下：

```
JFreeChart chart = getJFreeChart();
//获取多饼图
MultiplePiePlot multiplePiePlot = (MultiplePiePlot) chart.getPlot();
//获取多饼图 JFreeChart 对象
JFreeChart jFreeChart = multiplePiePlot.getPieChart();
PiePlot piePlot = (PiePlot) jFreeChart.getPlot();
```

设计过程

（1）创建 ChartUtil 类，编写 createDataset()方法，在该方法中设置图表数据，创建一个 CategoryDataset 对象。代码如下：

```
private static CategoryDataset createDataset() {
        double[ ][ ] data = new double[ ][ ] {
                    { 620, 410, 300 },
                    { 300, 390, 500 } };
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset(
                    "Dept",                     //行名称
                    "Month",                    //列名称
                    data);
        return dataset;
}
```

（2）创建 createPiePlot()方法，在该方法中获取 JFreeChart 的实例，然后使用 JFreeChart 的实例先后分别获取 MultiplePiePlot 实例、JFreeChart 实例和 PiePlot 实例，再通过修改相应的字体来解决乱码问题。代码如下：

```
public static void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        //窗体标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.BOLD, 20));
        //图例
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //获取多饼图
        MultiplePiePlot multiplePiePlot = (MultiplePiePlot) chart.getPlot();
        JFreeChart jFreeChart = multiplePiePlot.getPieChart();
        //图表标签
        PiePlot piePlot = (PiePlot) jFreeChart.getPlot();
        piePlot.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        piePlot.setLabelGenerator(new StandardPieSectionLabelGenerator("{0}:{1}"));
        //图表标题
        TextTitle textTitle2 = jFreeChart.getTitle();
        textTitle2.setFont(new Font("宋体", Font.BOLD, 20));
}
```

（3）创建 getJFreeChart()方法，在该方法中根据数据集生成多饼图的 JFreeChart 对象，并且指定多饼图排序方式为行排列。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = createDataset();      //获取数据集
        //生成 JFreeChart
        JFreeChart chart = ChartFactory.createMultiplePieChart(
                    "4-6 月销售排行",              //饼图标题
                    dataset,                      //数据集
                    TableOrder.BY_ROW,            //排序方式
                    true, true, false);
        return chart;
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 189：多饼图的乱码。

多饼图的结构实际分为两部分，一部分是窗体，另一部分是饼图。因此，使用解决普通饼图乱码的方法，不能完全解决多饼图的乱码问题。在窗体部分中分为窗体的标题和图示，在饼图部分中又分为饼图标题和标签，所以字体乱码问题也是从这几方面处理的。

实例 190	3D 多饼图 光盘位置：光盘\MR\08\190	高级 实用指数：★★★★

实例说明

普通饼图可以绘制成 3D 的效果，而多饼图同样可绘制成 3D 的效果。本实例将绘制一个具有 3D 效果的多饼图，如图 8.15 所示。

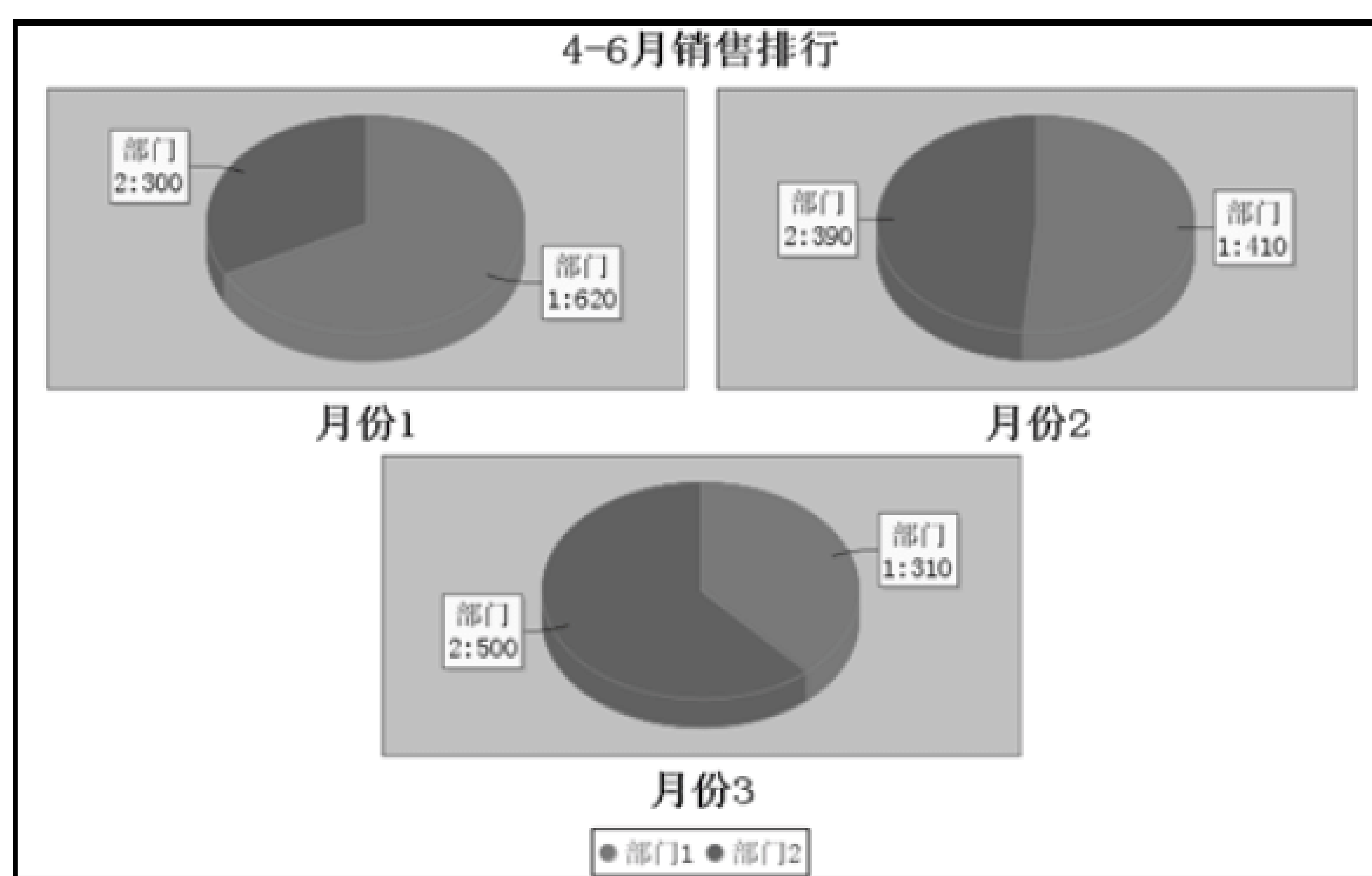


图 8.15　3D 多饼图

关键技术

JFreeChart 在 ChartFactory 类中提供了可以绘制 3D 多饼图的方法——createMultiplePieChart3D()。语法如下：

```
public static JFreeChart createMultiplePieChart3D(String title,CategoryDataset dataset, TableOrder order, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：表示 3D 多饼图的窗体的标题。

❷ dataset：表示 3D 多饼图的数据集合。

❸ order：设置 3D 多饼图的显示方式。

❹ legend：表示是否使用图示。

❺ tooltips：表示是否生成工具栏提示。

❻ urls：表示是否生成 URL 链接。

（1）创建 ChartUtil 类，编写 createDataset()方法，在该方法中设置图表数据，创建一个 CategoryDataset 对象。代码如下：

```
private static CategoryDataset createDataset() {
  double[ ][ ] data = new double[ ][ ] {
                { 620, 410, 300 },
                { 300, 390, 500 } };
//创建数据集合实例
  CategoryDataset dataset = DatasetUtilities.createCategoryDataset(
```

```
                    "部门",                              //行名称
                    "月份",                              //列名称
                    data);
    return dataset;
}
```

（2）创建 createPiePlot()方法，在该方法中获取 JFreeChart 的实例，然后使用 JFreeChart 的实例先后分别获取 MultiplePiePlot 实例、JFreeChart 实例和 PiePlot 实例，通过修改相应的字体来解决乱码问题。代码如下：

```
public static void createPiePlot(JFreeChart chart) {

        //窗体标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.BOLD, 20));
        //图例
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //获取多饼图
        MultiplePiePlot multiplePiePlot = (MultiplePiePlot) chart.getPlot();
        JFreeChart jFreeChart = multiplePiePlot.getPieChart();
        //图表标签
        PiePlot piePlot = (PiePlot) jFreeChart.getPlot();
        piePlot.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        piePlot.setLabelGenerator(new StandardPieSectionLabelGenerator("{0}:{1}"));
        //图表标题
        TextTitle textTitle2 = jFreeChart.getTitle();
        textTitle2.setFont(new Font("宋体", Font.BOLD, 20));
}
```

（3）创建 getJFreeChart()方法，在该方法中根据数据集生成多饼图的 JFreeChart 对象，并且指定 3D 饼形图效果。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = createDataset();
        JFreeChart chart = ChartFactory.createMultiplePieChart3D(
                "4-6 月销售排行  ",                      //饼图标题
                dataset,                                 //数据集
                TableOrder. BY_COLUMN,                   //排序方式
                true, true, false);
        createPiePlot(chart);
        return chart;
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表，具体代码参见配书光盘。

秘笈心法

心法领悟 190：设置 3D 多饼图的 Z 轴。

在 3D 的多饼图中，如果需要设置多饼图的 Z 轴高度，首先必须获取 MultiplePiePlot 对象，然后从 MultiplePiePlot 对象中获取 JFreeChart 对象，再从这个 JFreeChart 对象中获取 PiePlot3D 实例，使用 PiePlot3D 的 setDepthFactor()方法修改 Z 轴。

8.4　基本柱形图

实例 191	简单柱形图 光盘位置：光盘\MR\08\191	高级 实用指数：★★★

实例说明

本实例将使用 JFreeChart 提供的相应功能，绘制一个反映 2010 年上半年销售情况的简单柱形图，如图 8.16 所示。其中，X 轴即横轴，表示销售月份；Y 轴即竖轴，表示销售数量。

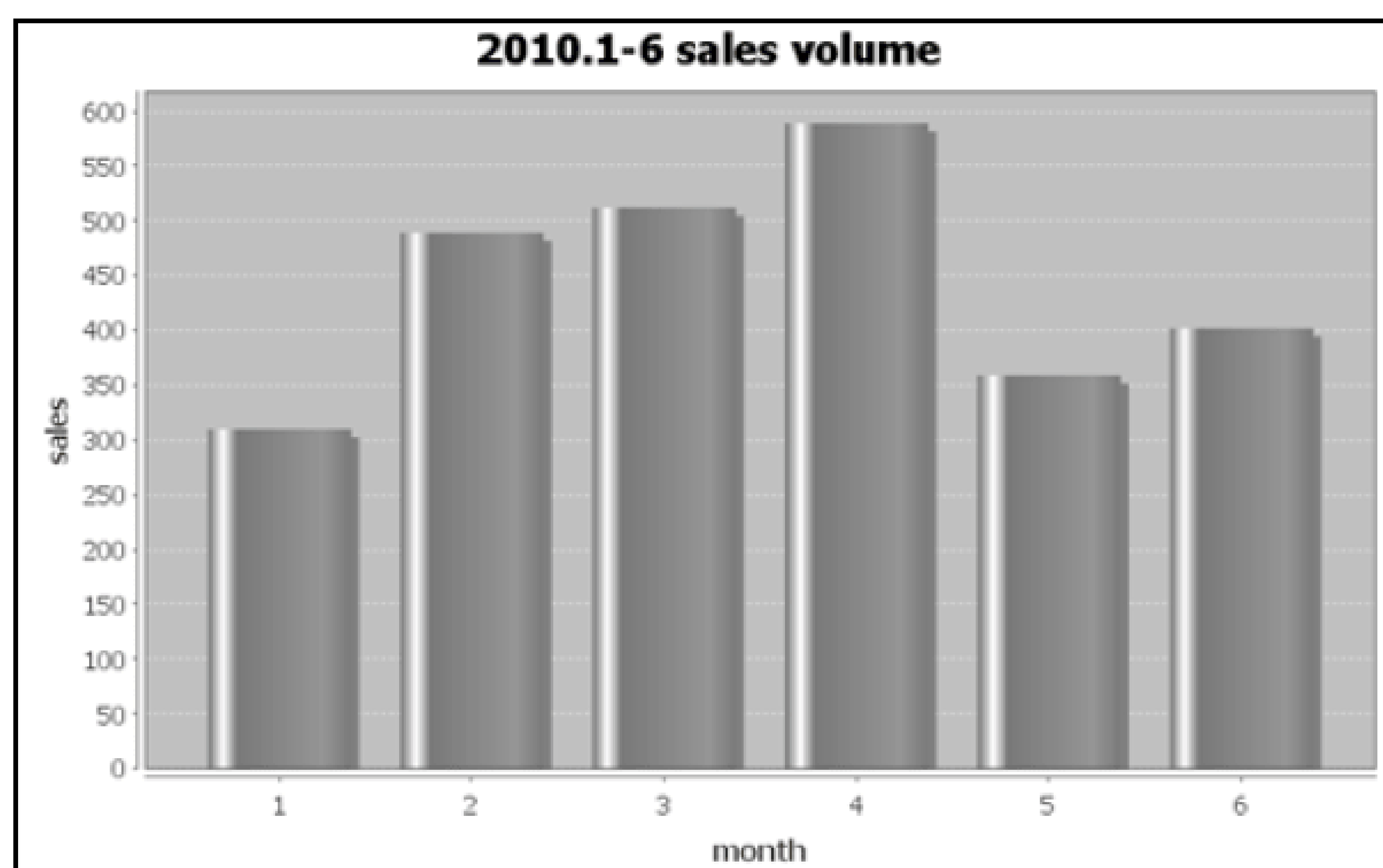


图 8.16　简单柱形图

关键技术

（1）利用 DatasetUtilities 类提供的 createCategoryDataset()方法，可以创建一个简单的柱形图数据集。语法如下：

```
public static CategoryDataset createCategoryDataset(Comparable rowKey, KeyedValues rowData)
```

参数说明

❶ rowKey：X 轴的含义，可以作为图示。

❷ rowData：数据集的内容，可以使用 DefaultKeyedValues 类创建数据集。

（2）利用 DefaultKeyedValues 类提供的 addValue()方法，可以为数据集添加数据。语法如下：

```
addValue(Comparable key, double value)
```

参数说明

❶ key：X 轴的名称。

❷ value：与 X 轴名称相对应的数值。

（3）JFreeChart 提供了创建普通柱形图的方法——createBarChart()，创建完成后会返回一个 JFreeChart 对象。语法如下：

```
public static JFreeChart createBarChart(String title, String categoryAxisLabel, String valueAxisLabel, CategoryDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：柱形图的标题。

❷ categoryAxisLabel：柱形图类别标签，即 X 轴的名称。

❸ valueAxisLabel：柱形图数据标签，即 Y 轴的名称。

❹ dataset：柱形图的数据集合。

❺ orientation：柱形图的图表方向。

❻ legend：表示是否使用图示。

❼ tooltips：表示是否生成工具栏提示。

❽ urls：表示是否生成 URL 链接。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1", 310);
        keyedValues.addValue("2", 489);
        keyedValues.addValue("3", 512);
```

```
        keyedValues.addValue("4", 589);
        keyedValues.addValue("5", 359);
        keyedValues.addValue("6", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法里获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010.1-6 sales volume",    //图表标题
                        "month",                                                   //X 轴标签
                        "sales",                                                   //Y 轴标签
                        dataset,                                                   //数据集
                        PlotOrientation.VERTICAL,                                  //图表方向：水平、垂直
                        false,                                                     //是否显示图例（对于简单的柱状图必须是 false）
                        false,                                                     //是否生成工具栏提示
                        false                                                      //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 191：柱形图的数据集。

创建柱形图的数据集时使用 DefaultKeyedValues 类。在使用 addValue()方法添加数据时，要注意先后顺序。添加数据集的顺序和 X 轴显示的顺序是一致的。

实例 192 柱形图角度

光盘位置：光盘\MR\08\192

高级

实用指数：★★★

实例说明

本实例将创建一个简单的 JFreeChart 柱形图，用以反映 2010 年上半年销售情况。为了丰富图表的展示效果，在此将柱形图横向展示出来，如图 8.17 所示。

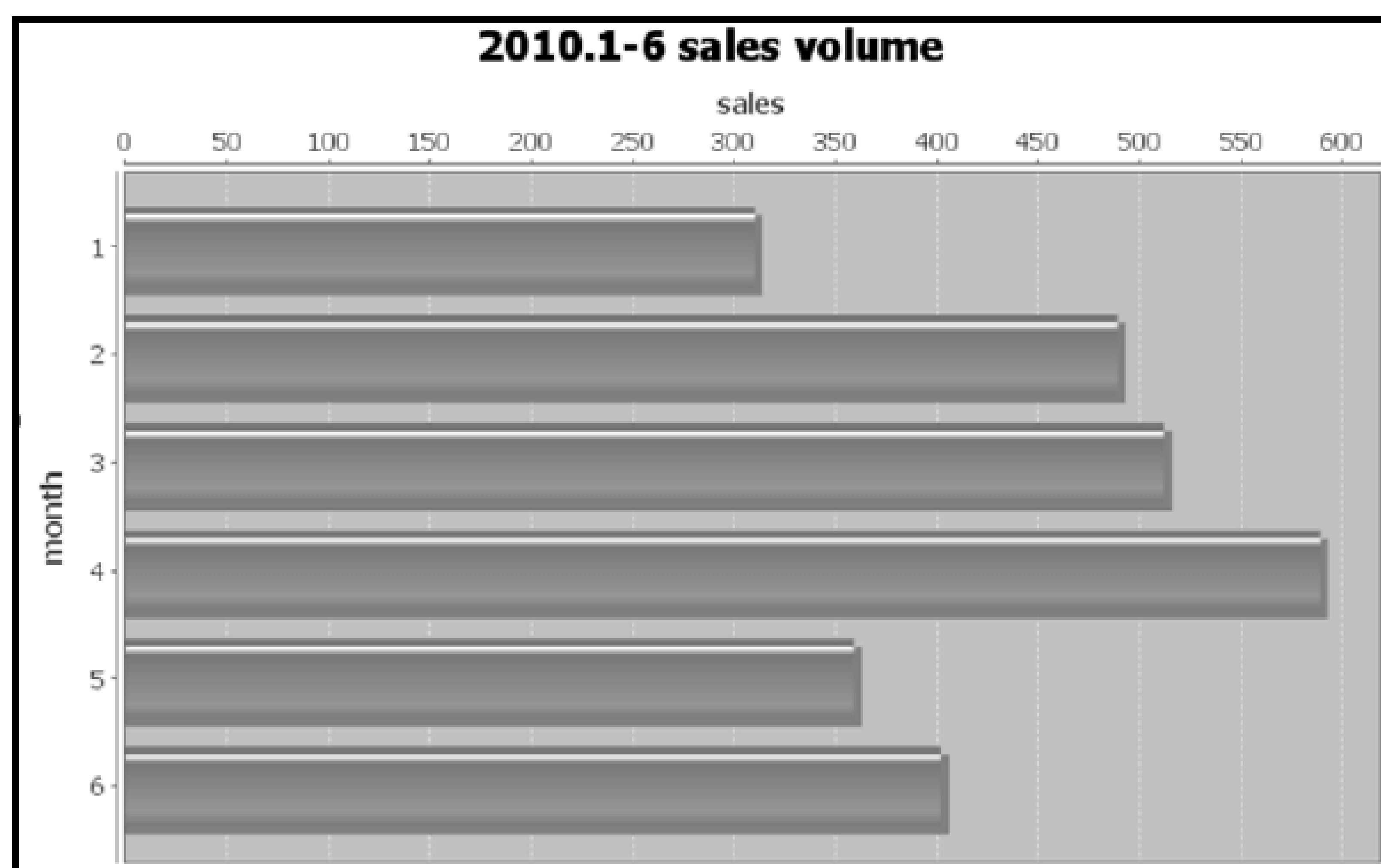


图 8.17　横向柱形图

JFreeChart 的 PlotOrientation 类定义了柱形图显示效果。

- PlotOrientation.HORIZONTAL：柱形图水平显示。
- PlotOrientation.VERTICAL：柱形图垂直显示。

使用方法如下：

```
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",  //图表标题
                "月份",                                             //X 轴标签
                "销售量（单位：本）",                                  //Y 轴标签
                dataset,                                            //数据集
                PlotOrientation. HORIZONTAL,                        //图表方向：水平、垂直
                false,                                              //是否显示图例（对于简单的柱形图必须是 false）
                false,                                              //是否生成工具栏提示
                false                                               //是否生成 URL 链接
                );
```

设计过程

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1", 310);
        keyedValues.addValue("2", 489);
        keyedValues.addValue("3", 512);
        keyedValues.addValue("4", 589);
        keyedValues.addValue("5", 359);
        keyedValues.addValue("6", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 createPlot()方法，在该方法中获取 JFreeChart 对象和 TextTitle 对象，修改标题字体。代码如下：

```
public static void createPlot(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
}
```

（3）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，同时设置柱形图水平显示。代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010.1-6 sales volume",  //图表标题
                "month",                                            //X 轴标签
                "sales",                                            //Y 轴标签
                dataset,                                            //数据集
                PlotOrientation. HORIZONTAL,                        //图表方向：水平、垂直
                false,                                              //是否显示图例（对于简单的柱形图必须是 false）
                false,                                              //是否生成工具栏提示
                false                                               //是否生成 URL 链接
                );
        createPlot(chart);                                          //修改标题字体
        return chart;
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

心法领悟 192：柱形图的角度。

柱形图的角度只有两种都使用 PlotOrientation 的常量，默认情况下使用 PlotOrientation.VERTICAL 常量表示柱形图以垂直的角度显示，还可以使用 PlotOrientation.HORIZONTAL 常量表示柱形图以水平的角度显示。



实例 193	柱形图负值 光盘位置：光盘\MR\08\193	高级 实用指数：★★★

实例说明

在数据的统计图表中，JFreeChart 的柱形图还支持负值或者空值的显示。本实例将实现柱形图的负值与空值的显示，运行结果如图 8.18 所示。

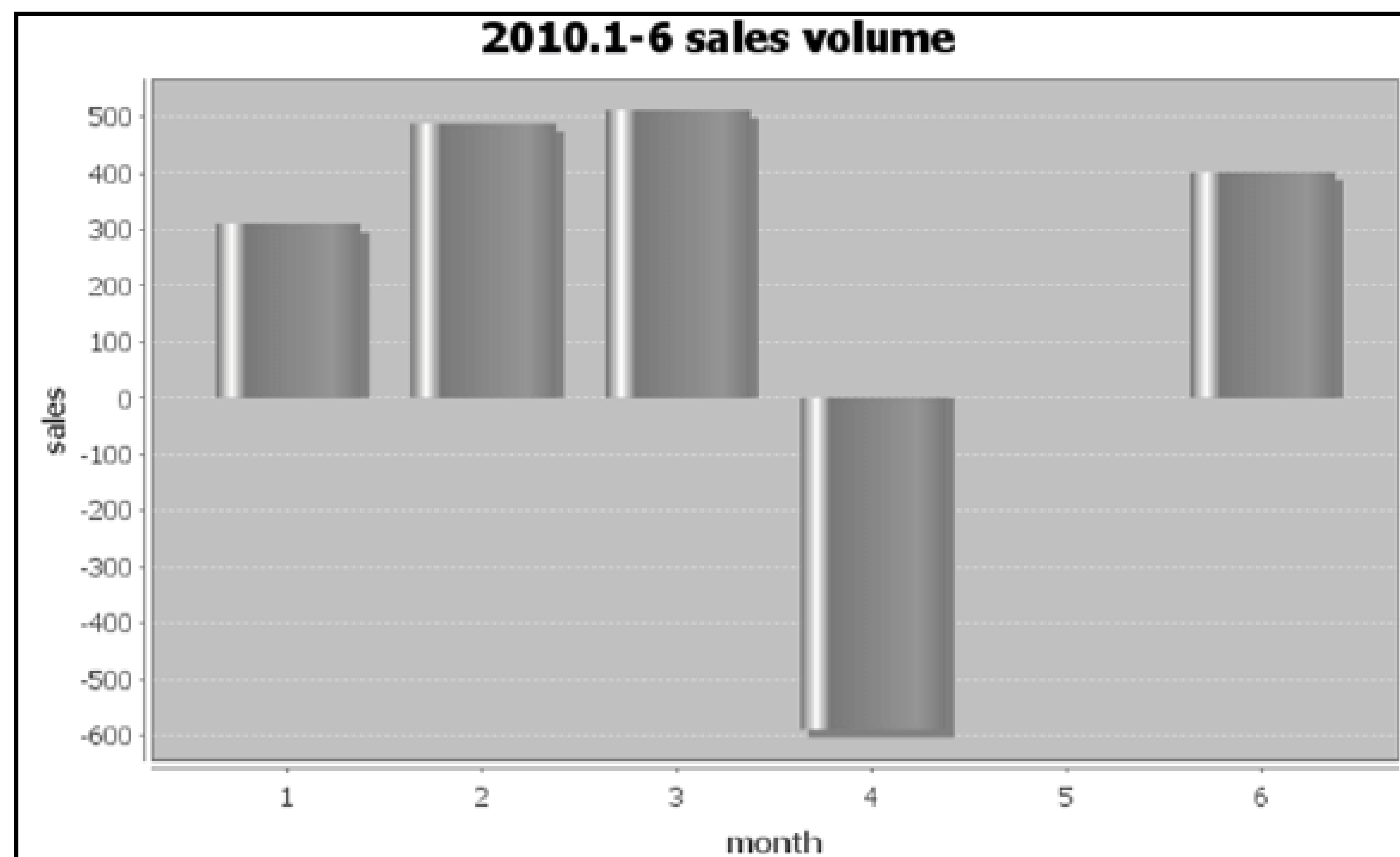


图 8.18　柱形图的负值

关键技术

JFreeChart 的 DefaultKeyedValues 类允许数据集接受负值。使用 addValue()方法可以向数据集添加数据。语法如下：

```
public void addValue(Comparable key, double value)
```

参数说明

❶ key：表示要添加的关键字。

❷ value：表示要添加的具体数据。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1", 310);
        keyedValues.addValue("2", 489);
        keyedValues.addValue("3", 512);
        keyedValues.addValue("4", -589);
        keyedValues.addValue("5", null);
        keyedValues.addValue("6", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010.1-6 sales volume",    //图表标题
                    "month",                                                       //X 轴标签
```

```
                "sales",                                    //Y 轴标签
                dataset,                                    //数据集
                PlotOrientation. HORIZONTAL,                //图表方向：水平、垂直
                false,                                      //是否显示图例（对于简单的柱形图必须是 false）
                false,                                      //是否生成工具栏提示
                false                                       //是否生成 URL 链接
                );
        return chart;
}
```

秘笈心法

心法领悟 193：柱形图的数据集。

柱形图的数据集中允许保存正数、负数、0 和空值，如果保存正数，则柱形图显示在 X 轴上方；如果保存负数，柱形图则显示在 X 轴下方；如果是 0 值或者是空值，柱形图则不会显示。

8.5　X 坐标轴

实例 194　X 轴字体　高级

光盘位置：光盘\MR\08\194　实用指数：★★★

实例说明

本实例将创建一个简单的 JFreeChart 柱形图，用以反映 2010 年上半年的销售情况。其中，在 X 轴上设置 1—6 月份的销售时间。在此要注意的是，如果使用默认字体，添加汉字时会产生乱码，而将其设置为宋体就不会有乱码出现，如图 8.19 所示。

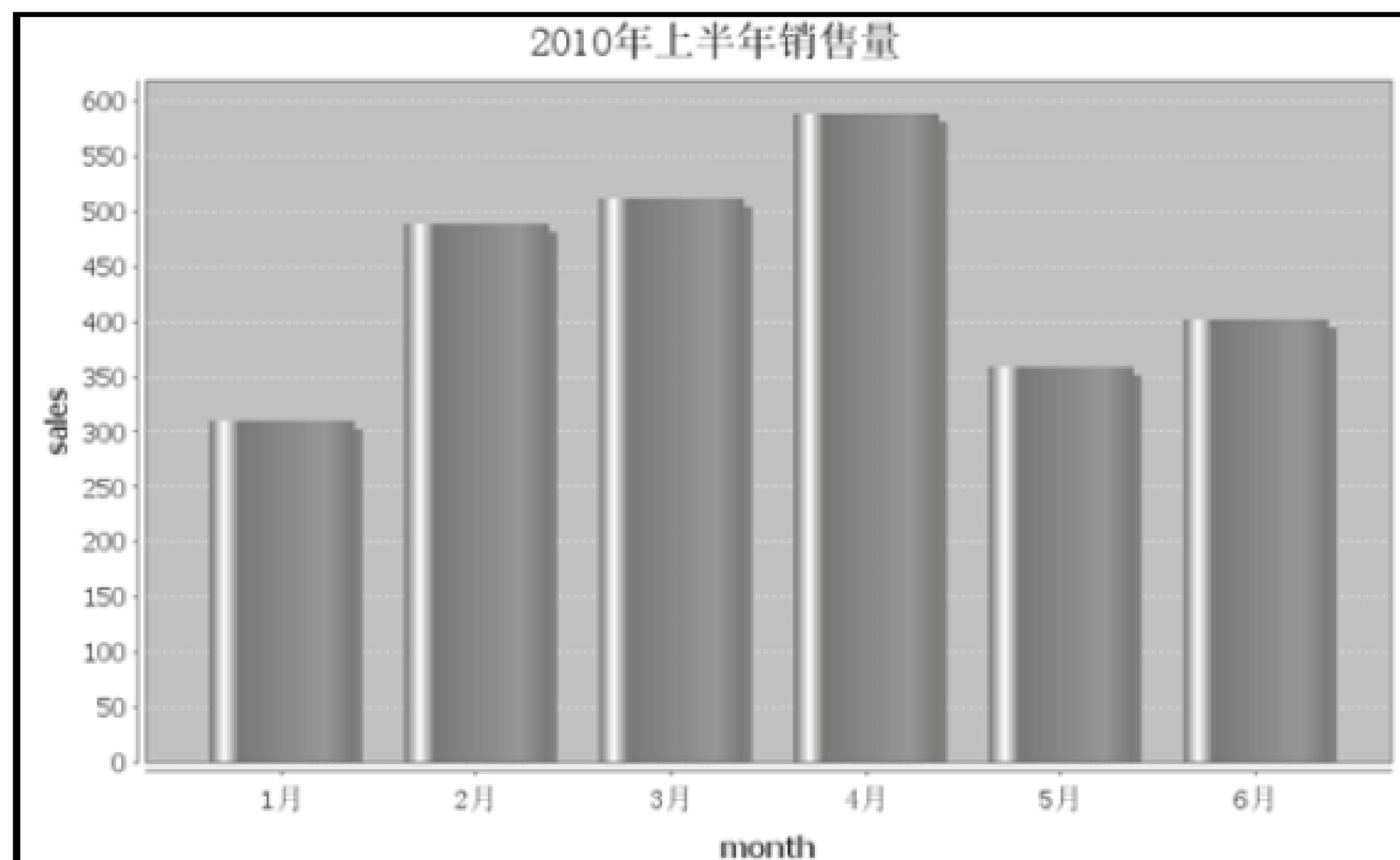


图 8.19　X 轴字体

使用 CategoryAxis 类的 setTickLabelFont()方法可以设置图表 X 轴的字体。语法如下：

```
public void setTickLabelFont(Font font)
```

参数说明

font：表示要设置的 X 轴的字体。

使用方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
CategoryAxis axis = categoryPlot.getDomainAxis();
//X 轴字体
axis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
```

设计过程

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "month",                                        //X 轴标签
                        "sales",                                        //Y 轴标签
                        dataset,                                        //数据集
                        PlotOrientation.VERTICAL,                       //图表方向：水平、垂直
                        false,                                          //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                          //是否生成工具栏提示
                        false                                           //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改标题和 X 轴的字体为“宋体”。代码如下：

```
public static void updateFont(JFreeChart chart){
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis axis = categoryPlot.getDomainAxis();
        //X 轴字体
        axis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 194：设置标题字体和 X 轴字体大小。

在一个图表中，一般情况下坐标轴的字体要比标题的字体小一些，所以本实例中设置标题字体大小为 20，而 X 轴的字体大小设置为 14。

实例 195	X 轴标签字体 光盘位置：光盘\MR\08\195	高级 实用指数：★★

X 轴的标签字体与 X 轴字体并不是同时设置的。本实例为 X 轴的标签设置宋体，使其能够支持汉字，效果如图 8.20 所示。

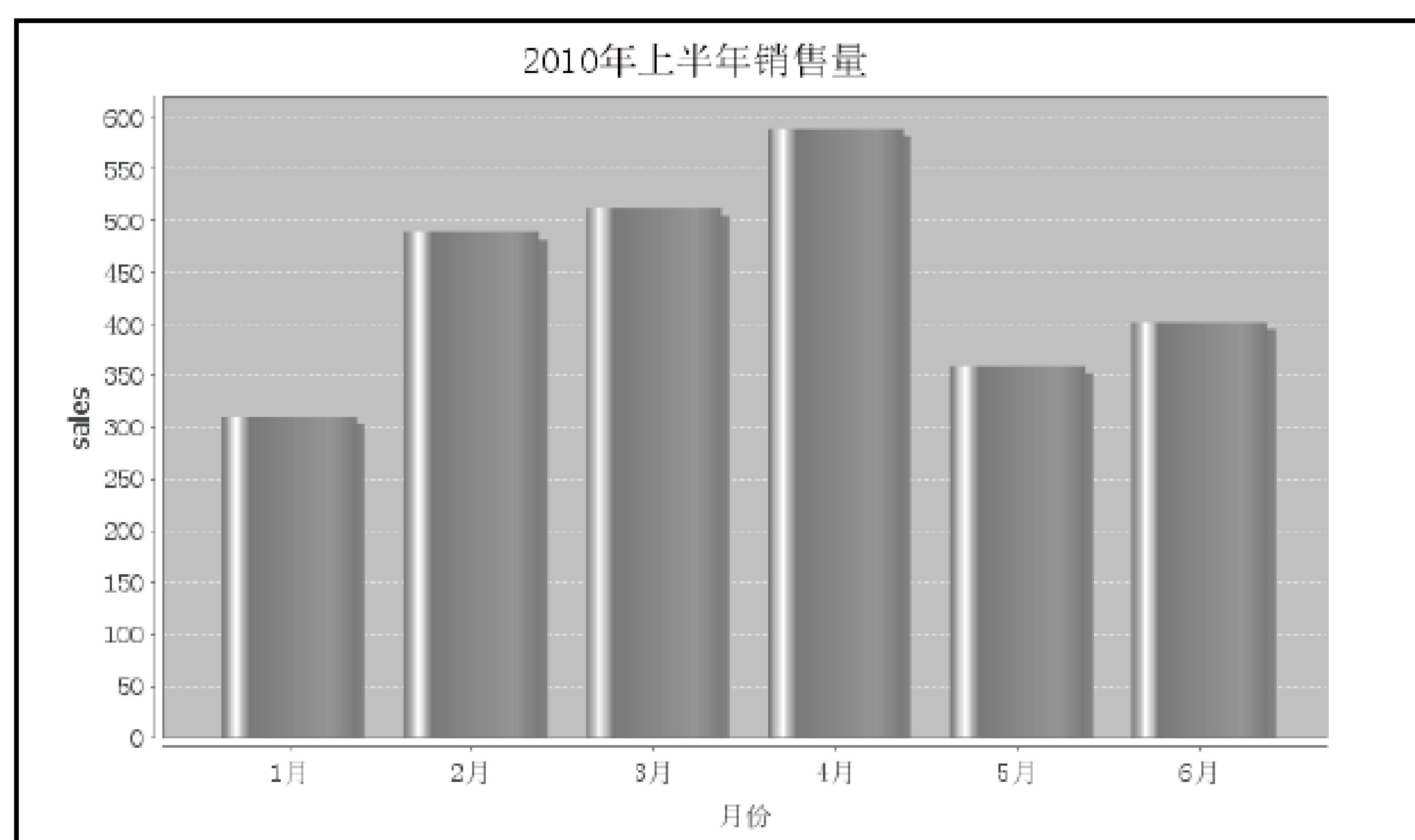


图 8.20　X 轴标签字体

关键技术

通过 JFreeChart 对象的 getCategoryPlot()方法可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，使用 CategoryAxis 的 setLabelFont()方法可以设置 X 轴标签的字体。语法如下：

```
public void setLabelFont(Font font)
```

参数说明

font：表示要设置的 X 轴的标签字体。

使用方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
CategoryAxis categoryAxis= categoryPlot.getDomainAxis();
//X 轴标签字体
categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
```


（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
private static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                "月份",                                                       //X 轴标签
                "sales",                                                      //Y 轴标签
                dataset,                                                      //数据集
                PlotOrientation.VERTICAL,                                     //图表方向：水平、垂直
                false,                                                        //是否显示图例（对于简单的柱形图必须是 false）
                false,                                                        //是否生成工具栏提示
                false                                                         //是否生成 URL 链接
                );
```

```
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改标题、X 轴、X 轴标签和 Y 轴的字体为“宋体”。代码如下：

```
public static void updateFont(JFreeChart chart){
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis= categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴字体
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 195：设置 X 轴标签字体与 X 轴上的字体。

X 轴标签字体与 X 轴上的字体虽然使用两个方法进行设置，但一般情况下两者的字体大小可以一致。例如，X 轴字体大小使用 14，X 轴标签的字体大小也可用 14。不过，也有很多用户喜欢把 X 轴标签的字体设置得比 X 轴上的字体稍微大一些。这些完全根据实际需要而定。

实例 196	X 轴标签角度 光盘位置：光盘\MR\08\196	高级 实用指数：★★★

实例说明

可以根据需要调整 X 轴标签的旋转角度，本实例将 X 轴的标签设置为垂直显示，如图 8.21 所示。

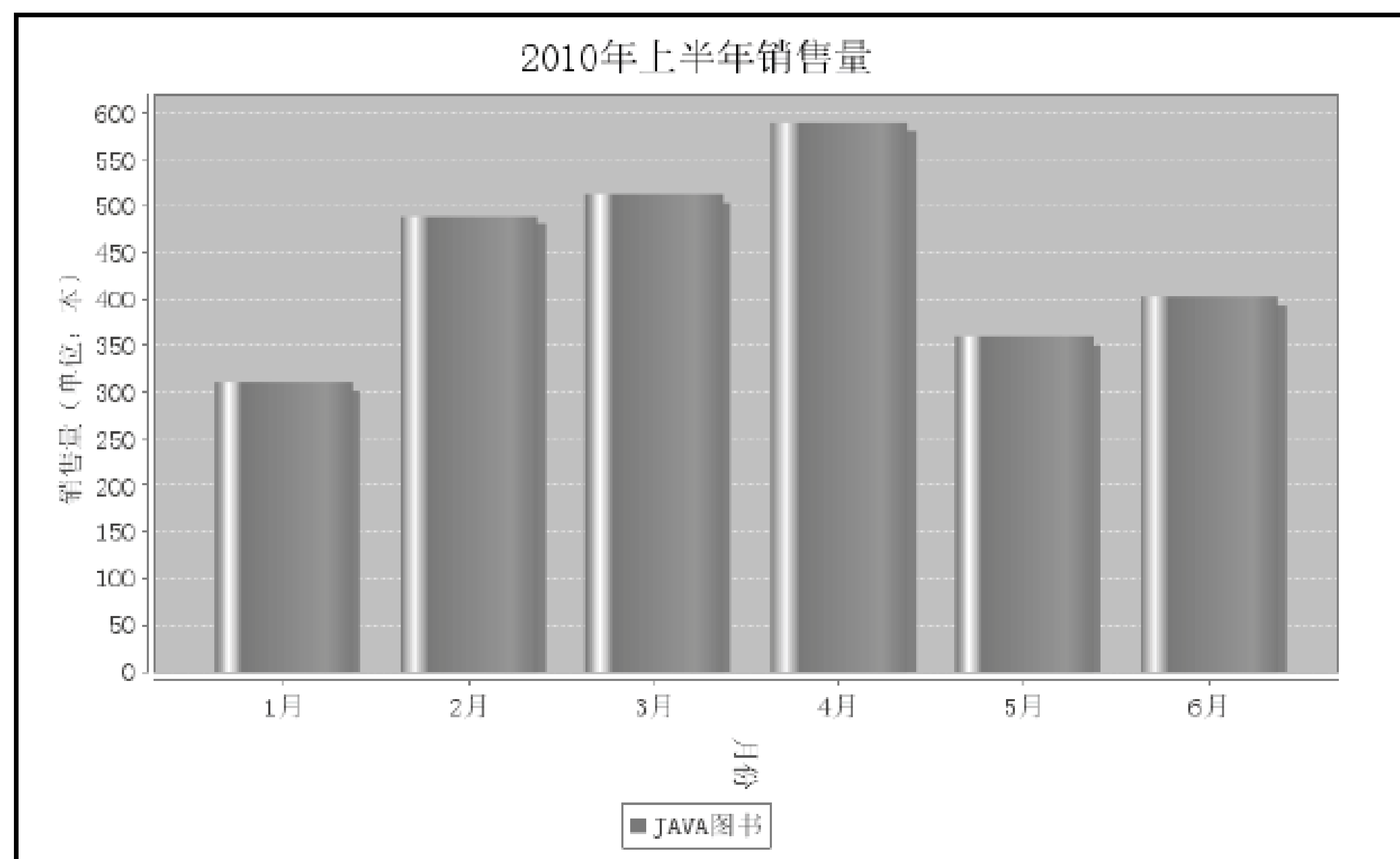


图 8.21　X 轴标签垂直显示

关键技术

使用 Axis 类的 setLabelAngle()方法可以设置 X 轴标签旋转的角度。语法如下：

```
public void setLabelAngle(double angle)
```

参数说明

angle：表示 X 轴标签旋转的弧度，根据弧度和 PI 值可以计算出 X 轴标签旋转的角度。

设计过程

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
	DefaultKeyedValues keyedValues = new DefaultKeyedValues();
	//添加数据
	keyedValues.addValue("1 月", 310);
	keyedValues.addValue("2 月", 489);
	keyedValues.addValue("3 月", 512);
	keyedValues.addValue("4 月", 589);
	keyedValues.addValue("5 月", 359);
	keyedValues.addValue("6 月", 402);
	//创建数据集合实例
	CategoryDataset dataset = DatasetUtilities.createCategoryDataset("JAVA 图书", keyedValues);
	return dataset;
}
```

（2）编写 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
private sttic JFreeChart getJFreeChart() {
	CategoryDataset dataset = getCategoryDataset();
	JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",	//图表标题
			"月份",							//X 轴标签
			"销售量（单位：本）",					//Y 轴标签
			dataset,							//数据集
			PlotOrientation.VERTICAL,				//图表方向：水平、垂直
			true,							//是否显示图例（对于简单的柱形图必须是 false）
			false,							//是否生成工具栏提示
			false							//是否生成 URL 链接
			);
	return chart;
}
```

（3）编写 updateFont()方法，在该方法中修改标题、X 轴、Y 轴、X 轴标签、Y 轴标签的字体为“宋体”。代码如下：

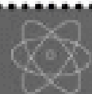
```
private static void updateFont(JFreeChart chart) {
	//标题
	TextTitle textTitle = chart.getTitle();
	textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
	LegendTitle legendTitle = chart.getLegend();
	legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
	//图表
	CategoryPlot categoryPlot = chart.getCategoryPlot();
	CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
	//X 轴字体
	categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
	//X 轴标签字体
	categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
	ValueAxis valueAxis = categoryPlot.getRangeAxis();
	//Y 轴字体
	valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
	//Y 轴标签字体
	valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

心法领悟 196：Axis 抽象类。

Axis 是一个抽象类，Y 轴与 X 轴分别使用 ValueAxis 和 CategoryAxis 继承了这个抽象类，所以 Y 轴与 X 轴设置标签的旋转角度相同，都是使用 Axis 类的 setLabelAngle()方法，不同的是 X 轴使用 CategoryAxis 的实例，

而 Y 轴使用 ValueAxis 实例。



实例 197　X 轴尺度线颜色

光盘位置：光盘\MR\08\197

高级　实用指数：★★★

实例说明

在 JFreeChart 的柱形图中，X 轴尺度线的颜色是可以改变的。本实例将把 X 轴的尺度线改成绿色，运行结果如图 8.22 所示。

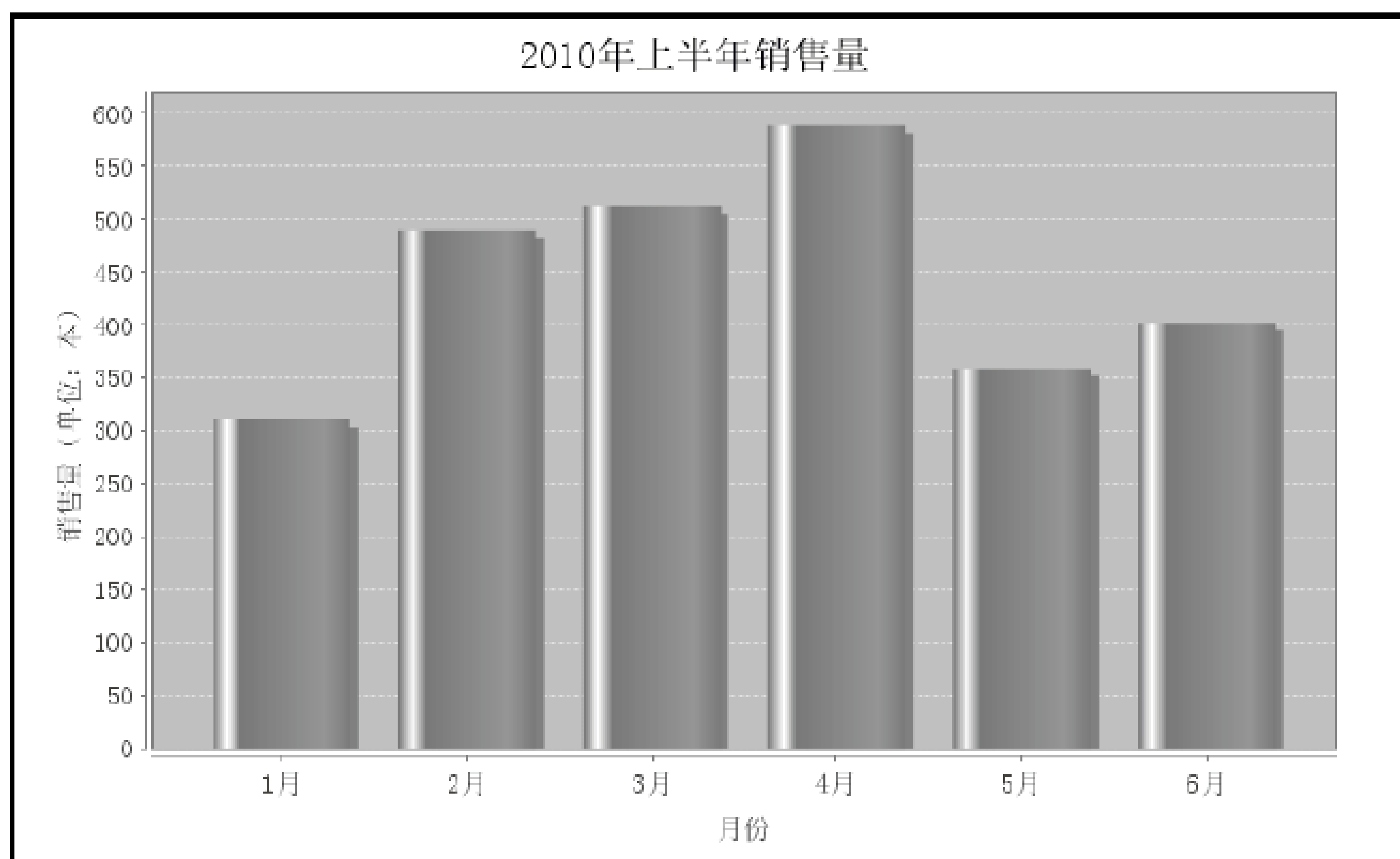


图 8.22　X 轴尺度线颜色

关键技术

通过 JFreeChart 对象的 getCategoryPlot()方法可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，使用 CategoryAxis 对象的 setAxisLinePaint()方法可以为 X 轴尺度线设置颜色。语法如下：

```
public void setAxisLinePaint(Paint paint)
```

参数说明

paint：表示 X 轴尺度线设置的颜色。

使用方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
//X 轴（分类轴）
CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
//X 轴尺度线颜色
categoryAxis.setAxisLinePaint(Color.GREEN);
```


（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合，代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
```

```
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）编写 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "月份",                                    //X 轴标签
                        "销售量（单位：本）",                      //Y 轴标签
                        dataset,                                   //数据集
                        PlotOrientation.VERTICAL,                  //图表方向：水平、垂直
                        false,                                     //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                     //是否生成工具栏提示
                        false                                      //是否生成 URL 链接
                        );
        return chart;
}
```

（3）编写 updatePlot()方法，将 X 轴尺度线颜色设置为绿色。代码如下：

```
private static void updatePlot(JFreeChart chart){
        CategoryPlot categoryPlot = chart.getCategoryPlot();                  //图表（柱形图）
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();             //X 轴（分类轴）
        categoryAxis.setAxisLinePaint(Color.GREEN);                           //X 轴尺度线颜色
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 197：X 轴尺度线的颜色。

在 JFreeChart 的图表中，X 轴的尺度线颜色默认为灰色，在 Java 的 Color 类中的常量为 Color.GRAY。本实例设置 X 轴尺度线为绿色时，使用了 Color 常量 Color.GREEN。如果希望修改 X 轴字体的颜色，可以使用 CategoryAxis 类的 setTickLabelPaint()方法。

实例 198　隐藏 X 轴尺度线

光盘位置：光盘\MR\08\198

高级

实用指数：★★★

X 轴的尺度线为在图表的 X 轴下方多出的一条有刻度的线，在使用时可以根据需要将其去掉。本实例将演示如何去掉 X 轴的尺度线，运行结果如图 8.23 所示。

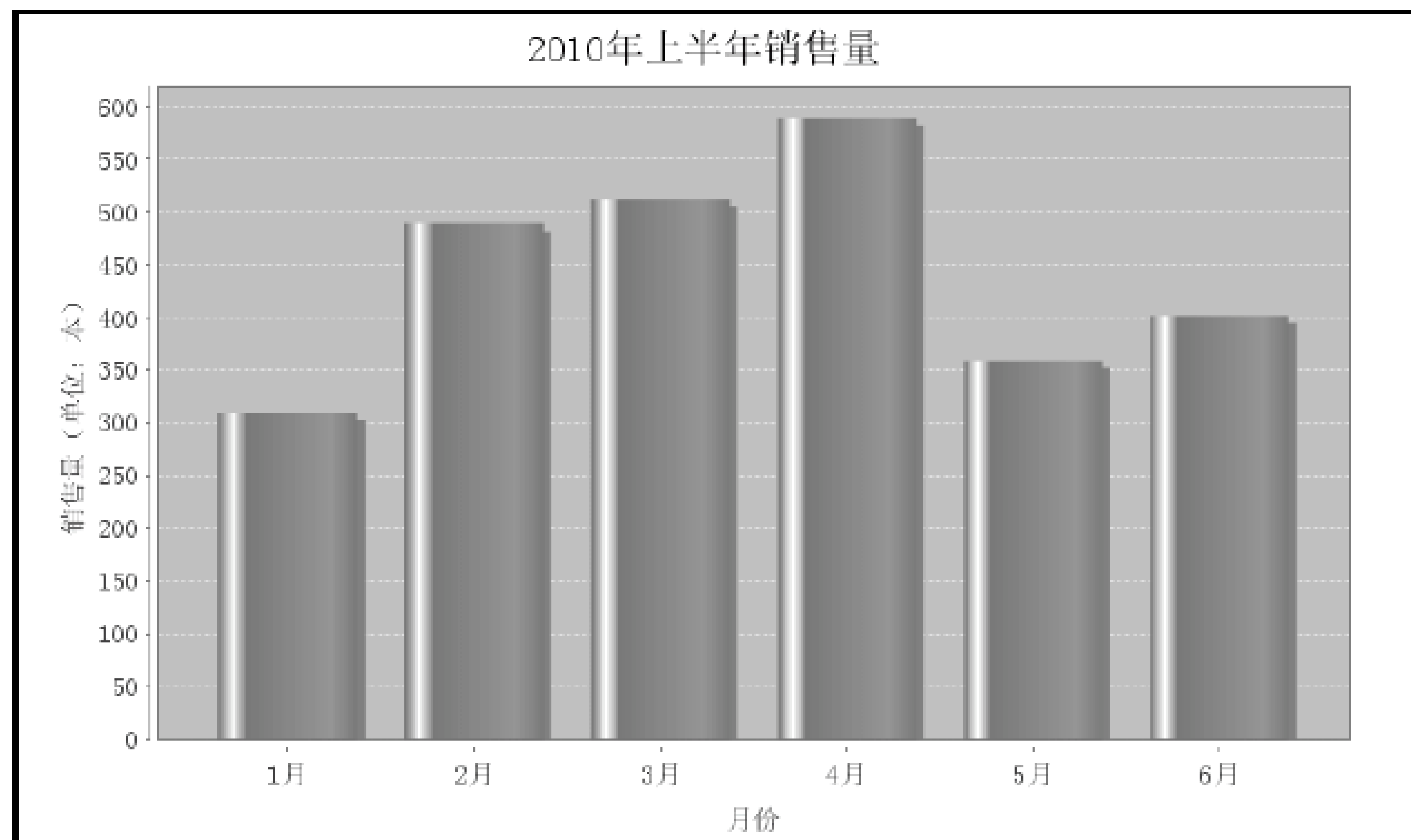


图 8.23　隐藏 X 轴尺度线

关键技术

通过 JFreeChart 对象的 getCategoryPlot()方法可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，使用 CategoryAxis 对象的 setAxisLineVisible()方法可以隐藏 X 轴尺度线。语法如下：

```
public void setAxisLineVisible(boolean visible)
```

参数说明

visible：表示 X 轴尺度线是否显示。当 visible 为 true 时，显示尺度线；当 visible 为 false 时，则不显示尺度线。

使用方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
//X 轴
CategoryAxis axis = categoryPlot.getDomainAxis();
//尺度线
axis.setAxisLineVisible(false);
```

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
private static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "月份",                                         //X 轴标签
                        "销售量（单位：本）",                           //Y 轴标签
                        dataset,                                        //数据集
                        PlotOrientation.VERTICAL,                       //图表方向：水平、垂直
                        false,                                          //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                          //是否生成工具栏提示
                        false                                           //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中隐藏 X 轴尺度线。代码如下：

```
private static void updatePlot(JFreeChart chart){
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //X 轴
        CategoryAxis axis = categoryPlot.getDomainAxis();
        //尺度线
        axis.setAxisLineVisible(false);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 198：X 轴的尺度线。

X 轴的尺度线在默认情况下是显示的，使用者可以根据需要决定是否使用尺度线。如果 X 轴的尺度线被隐藏，尺度线上的刻度会自动被显示在图表的下边框上。



实例 199	X 轴尺度线笔触 光盘位置：光盘\MR\08\199	高级 实用指数：★★

实例说明

X 轴的尺度线可以根据实际情况进行调整，例如，将其加粗、修改其样式。本实例把 X 轴的尺度线加粗到 5，运行结果如图 8.24 所示。

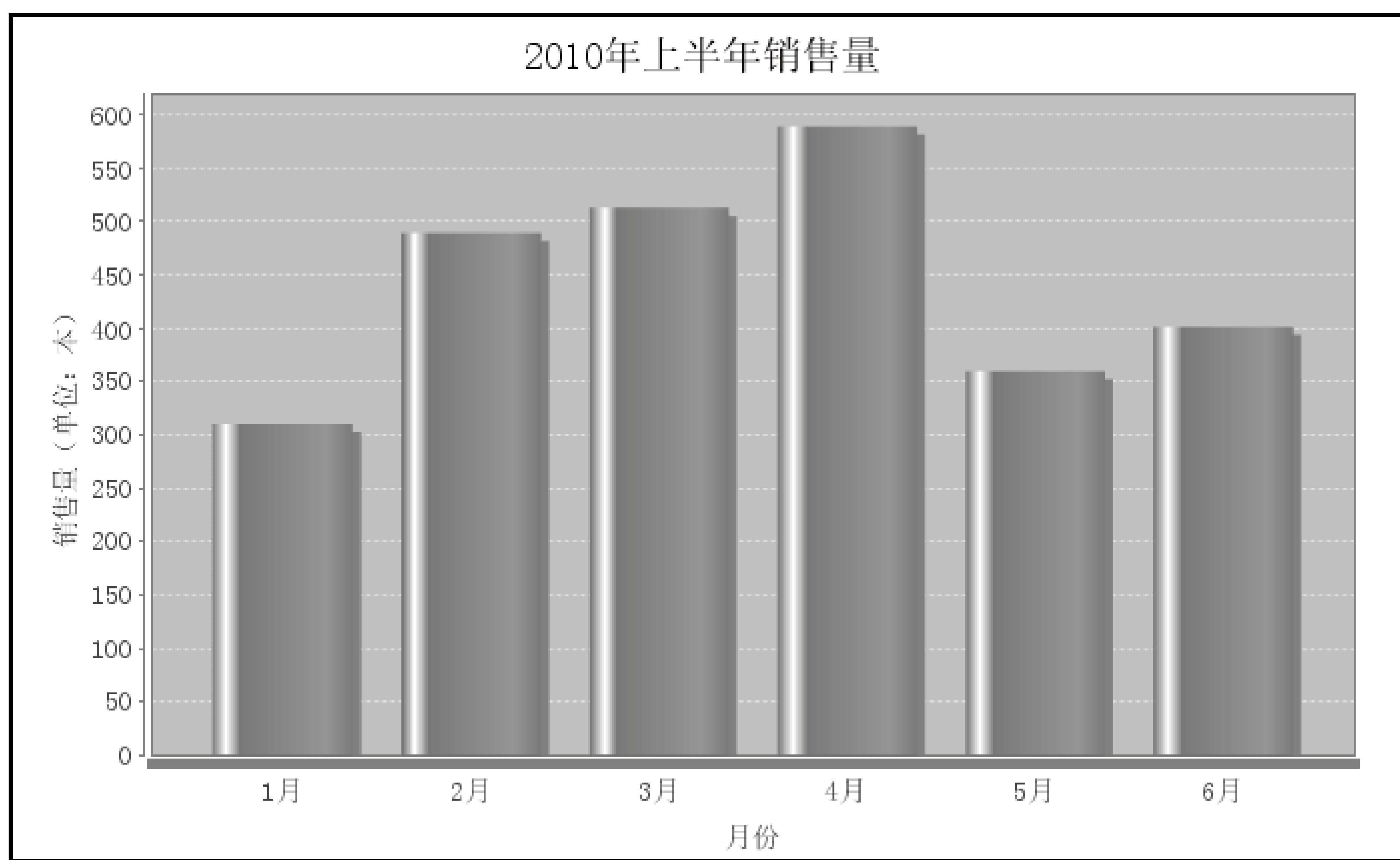


图 8.24　加粗 X 轴尺度线笔触

关键技术

通过 JFreeChart 对象的 getCategoryPlot()方法可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，在 CategoryAxis 对象中使用如下方法可以加粗 X 轴尺度线。

```
public BasicStroke(float width)
```

参数说明

width：表示笔触要加粗的数值。

使用方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
//X 轴
CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
//尺度线笔触
categoryAxis.setAxisLineStroke(new BasicStroke(5));
```

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
```

```
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
private static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "月份",                                          //X 轴标签
                        "销售量（单位：本）",                              //Y 轴标签
                        dataset,                                         //数据集
                        PlotOrientation.VERTICAL,                        //图表方向：水平、垂直
                        false,                                           //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                           //是否生成工具栏提示
                        false                                            //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改标题、X 轴、Y 轴、X 轴标签、Y 轴标签的字体为“宋体”。代码如下：

```
public static void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

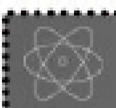
（4）创建 updatePlot()方法，在该方法中加粗 X 轴尺度线。代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //X 轴字体
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //尺度笔触
        categoryAxis.setAxisLineStroke(new BasicStroke(5));
}
```

（5）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

心法领悟 199：X 轴尺度线的笔触。

X 轴尺度线的笔触默认值为 1，用户可以根据实际需要决定使用多大数值的笔触。如果 X 轴尺度线被设置为隐藏状态，那么尺度线笔触的设置也就没有必要。



实例 200　X 轴尺度标签

光盘位置：光盘\MR\08\200

高级

实用指数：★★★

实例说明

当 X 轴的尺度之间距离有限时，如果标签名称太长，相邻两个标签的文字可能会重叠。本实例将把轴的标签显示方向进行调整，即旋转一定角度，这样就避免了这种情况的发生，效果如图 8.25 所示。

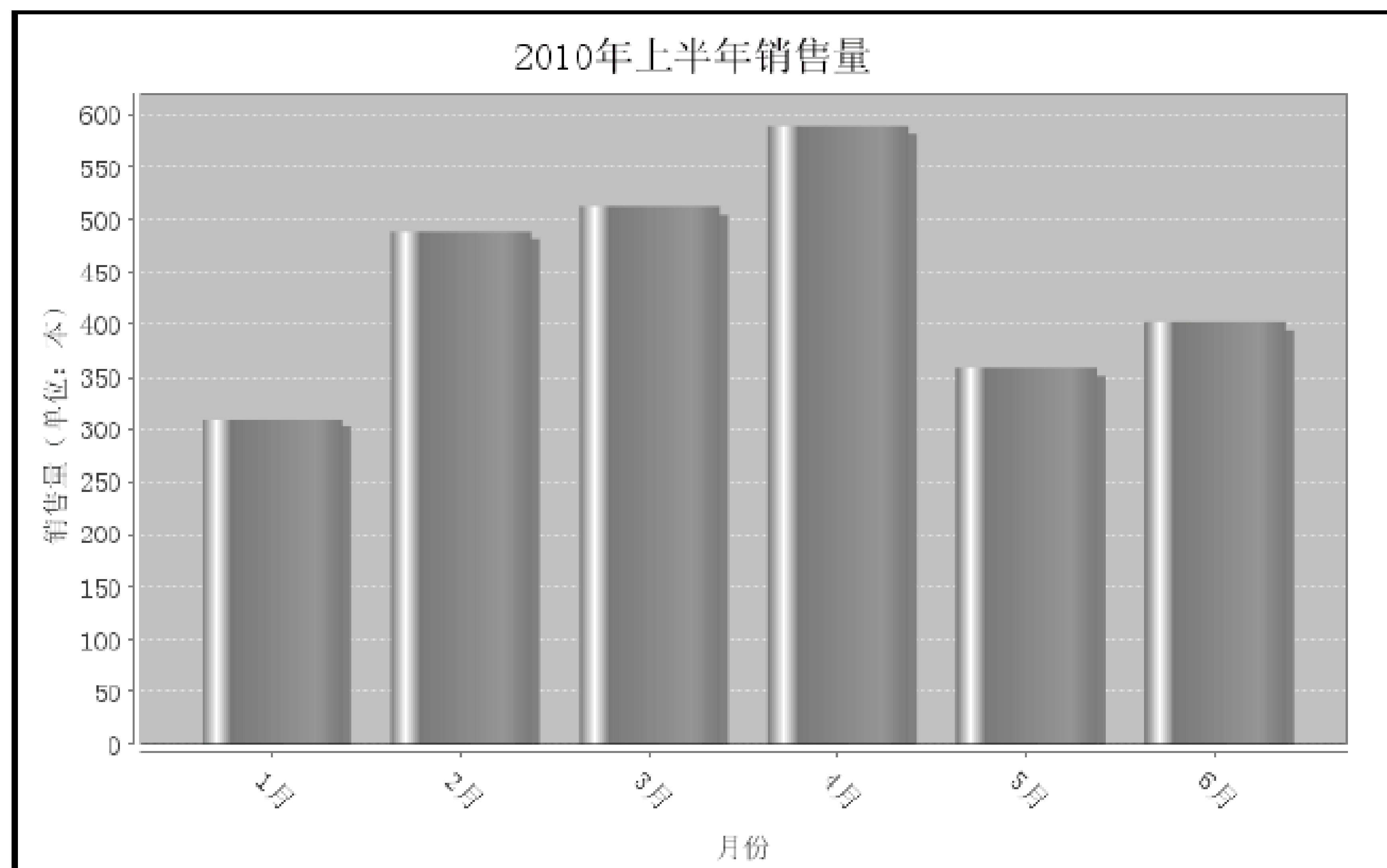


图 8.25　旋转 X 轴尺度标签

关键技术

通过 JFreeChart 对象的 getCategoryPlot()方法可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，在 CategoryAxis 对象中使用如下方法可以修改 X 轴尺度标签的角度。

```
public void setCategoryLabelPositions(CategoryLabelPositions positions)
```

参数说明

positions：表示要修改的尺度标签的角度，必须使用 CategoryLabelPositions 类来实现。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
```

```
                    "月份",                                                              //X 轴标签
                    "销售量（单位：本）",                                                //Y 轴标签
                    dataset,                                                             //数据集
                    PlotOrientation.VERTICAL,                                            //图表方向：水平、垂直
                    false,                                                               //是否显示图例（对于简单的柱形图必须是 false）
                    false,                                                               //是否生成工具栏提示
                    false                                                                //是否生成 URL 链接
                    );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中把 X 轴尺度标签顺时针旋转 45°。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //X 轴
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴尺度标签
        categoryAxis.setCategoryLabelPositions(CategoryLabelPositions.DOWN_45);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 200：设置分类轴尺度标签的角度。

CategoryLabelPositions 类中定义了如下几个常用的常量，用来表示不同的角度。

- DOWN_45：表示分类轴尺度标签顺时针旋转 45°。
- DOWN_90：表示分类轴尺度标签顺时针旋转 90°。
- UP_45：表示分类轴尺度标签逆时针旋转 45°。
- UP_90：表示分类轴尺度标签逆时针旋转 90°。

实例 201 X 轴分类的间距

光盘位置：光盘\MR\08\201

高级

实用指数：★★

实例说明

JFreeChart 自动生成的柱形图中，X 轴的每个分类柱的宽度是由分类的数量决定的。如果柱形图的分类比较少，那么柱形的宽度就会很宽，图表看起来很不协调。本实例通过调整分类之间的间隔使整个图表更美观，如图 8.26 所示。

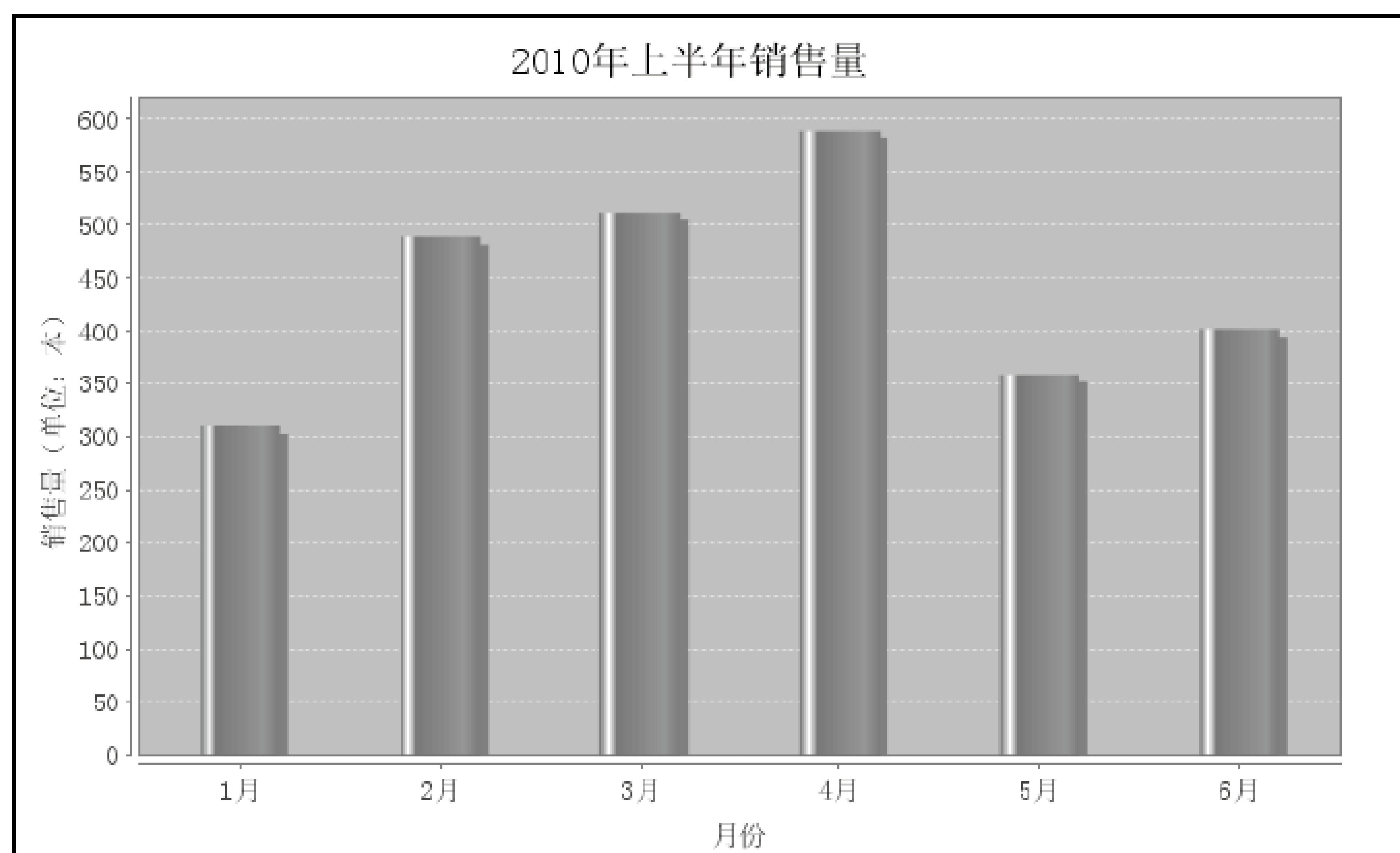


图 8.26　X 轴分类的间距

关键技术

通过 JFreeChart 对象的 getCategoryPlot 方法可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，在 CategoryAxis 对象中使用如下方法可以调整 X 轴分类的间距。

```
public void setCategoryMargin(double margin)
```

参数说明

margin：X 轴分类之间的距离，数值越大，间距越大。

设计过程

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
	DefaultKeyedValues keyedValues = new DefaultKeyedValues();
	//添加数据
	keyedValues.addValue("1 月", 310);
	keyedValues.addValue("2 月", 489);
	keyedValues.addValue("3 月", 512);
	keyedValues.addValue("4 月", 589);
	keyedValues.addValue("5 月", 359);
	keyedValues.addValue("6 月", 402);
	//创建数据集合实例
	CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
	return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
	CategoryDataset dataset = getCategoryDataset();
	JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",	//图表标题
			"月份",						//X 轴标签
			"销售量（单位：本）",				//Y 轴标签
			dataset,					//数据集
			PlotOrientation.VERTICAL,			//图表方向：水平、垂直
			false,						//是否显示图例（对于简单的柱形图必须是 false）
			false,						//是否生成工具栏提示
			false						//是否生成 URL 链接
			);
	return chart;
}
```

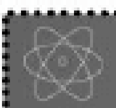
（3）创建 updatePlot()方法，在该方法中把 X 轴分类之间的距离设为 0.5。代码如下：

```
public static void updatePlot(JFreeChart chart){
	// 图表
	CategoryPlot categoryPlot = chart.getCategoryPlot();
	//X 轴
	CategoryAxis axis = categoryPlot.getDomainAxis();
	//X 轴分类间距
	axis.setCategoryMargin(0.5);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

心法领悟 201：setCategoryMargin()方法。

setCategoryMargin()方法用于设置 X 轴分类的间距，JFreeChart 默认的间距为 0.2。数值越小，分类之间的距离越小，则柱形的宽度越宽；反之，数值越大，分类之间的距离越大，则柱形的宽度越窄。



实例 202 X 轴分类与原点的间距

光盘位置：光盘\MR\08\202

高级

实用指数：★★★

实例说明

X 轴上的柱形可以根据实际需要向左或向右进行调整，本实例把 X 轴上的柱形向右作了调整，运行结果如图 8.27 所示。

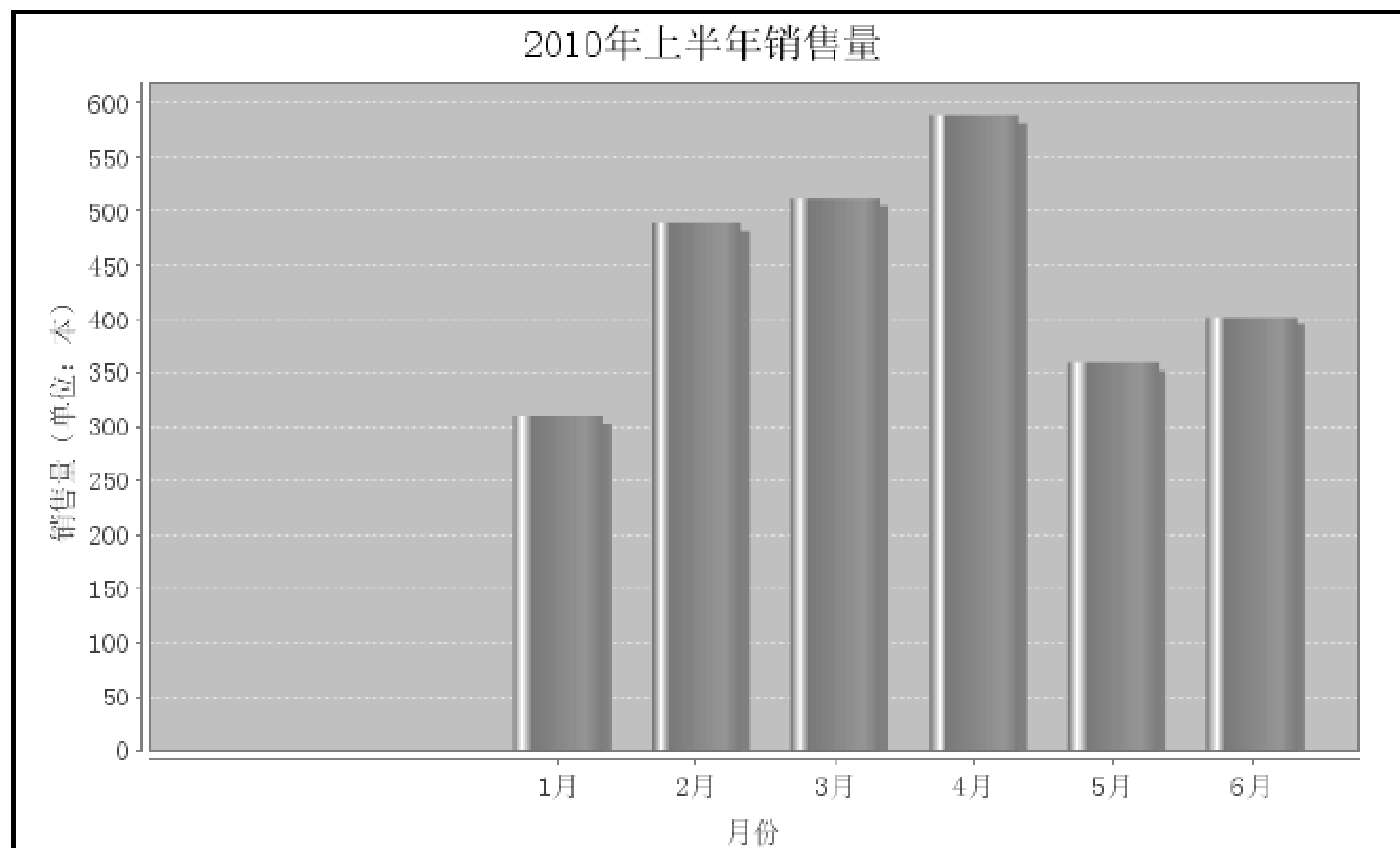


图 8.27　向右移动柱形

关键技术

通过 JFreeChart 对象的 getCategoryPlot()方法可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，在 CategoryAxis 对象中使用如下方法可以修改 X 轴上的柱形与坐标轴原点的距离。

```
public void setLowerMargin(double margin)
```

参数说明

margin：表示 X 轴上的柱形与坐标轴原点的距离，数值越大，距离原点越远。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
	DefaultKeyedValues keyedValues = new DefaultKeyedValues();
	//添加数据
	keyedValues.addValue("1 月", 310);
	keyedValues.addValue("2 月", 489);
	keyedValues.addValue("3 月", 512);
	keyedValues.addValue("4 月", 589);
	keyedValues.addValue("5 月", 359);
	keyedValues.addValue("6 月", 402);
	//创建数据集合实例
	CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
	return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
	CategoryDataset dataset = getCategoryDataset();
	JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",  //图表标题
```

```
                    "月份",                                  //X 轴标签
                    "销售量（单位：本）",                    //Y 轴标签
                    dataset,                                 //数据集
                    PlotOrientation.VERTICAL,                //图表方向：水平、垂直
                    false,                                   //是否显示图例（对于简单的柱形图必须是 false）
                    false,                                   //是否生成工具栏提示
                    false                                    //是否生成 URL 链接
                    );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中把 X 轴上的柱形与坐标原点的距离设置为 0.3。代码如下：

```
public static void updatePlot(JFreeChart chart){
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //X 轴
        CategoryAxis axis = categoryPlot.getDomainAxis();
        //与原点的距离
        axis.setLowerMargin(0.3);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 202：柱形的显示位置。

X 轴上的柱形与坐标原点的默认距离为 0.05，本实例向右移动了柱形，同时也缩短了柱形在图表上的展示长度，那么柱形的宽度也会相应地变窄。此外，还可以让柱形集体向原点靠拢，可以使用 CategoryAxis 的 setUpperMargin()方法来实现。语法如下：

```
setUpperMargin(double margin)
```

参数说明

margin：表示柱形与 X 轴在图表上终点的距离。

实例 203　X 轴的显示位置

光盘位置：光盘\MR\08\203

高级　实用指数：★★★

一般的柱形图都是垂直显示，X 轴位于图表的下方，但有时根据需要也可能会对 X 轴进行调整。本实例将演示如何将 X 轴显示在图表上方，如图 8.28 所示。

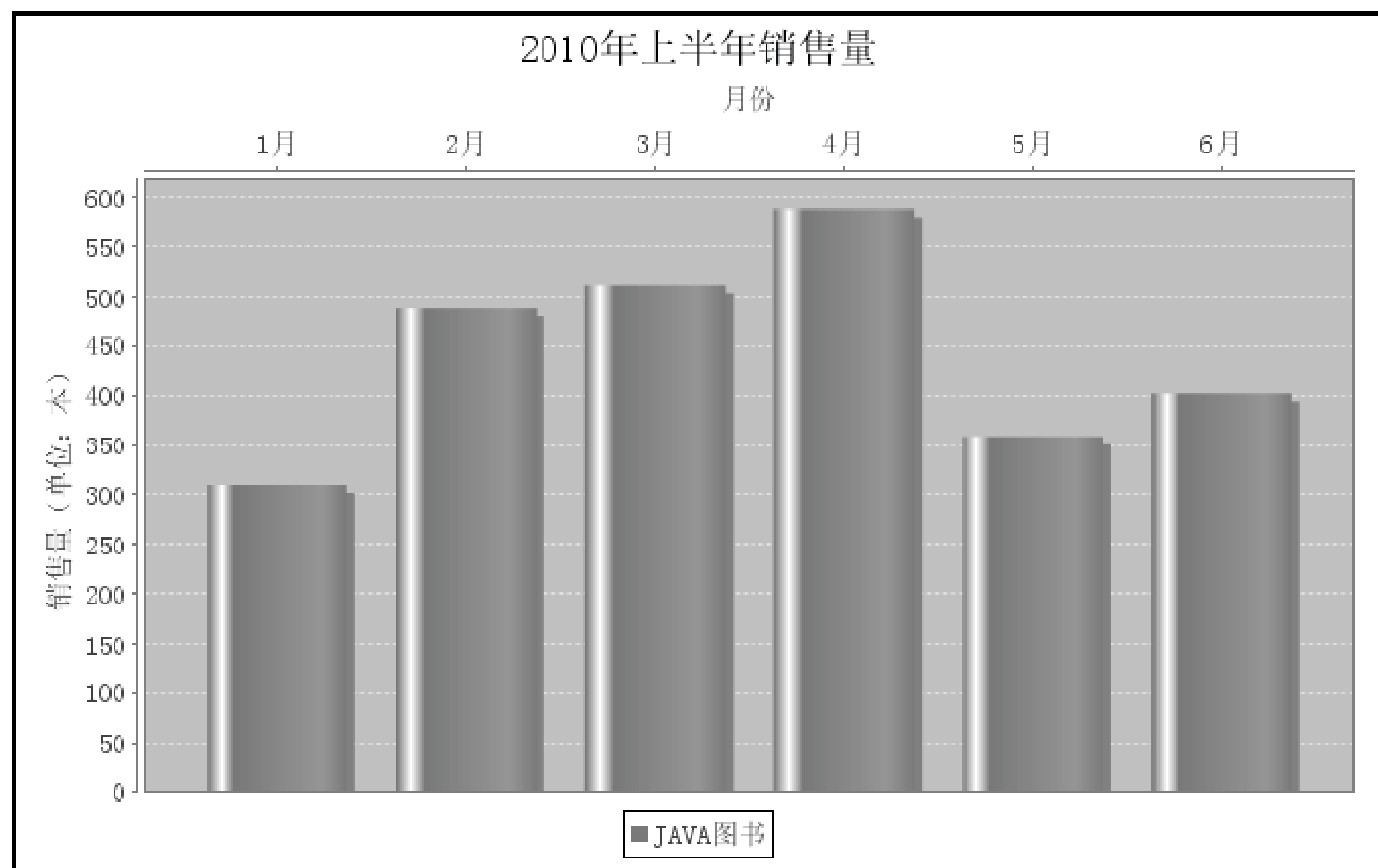


图 8.28　设置 X 轴显示位置

关键技术

在 CategoryPlot 中使用 setDomainAxisLocation()方法可以设置 X 轴在图表中的显示位置。语法如下：

```
public void setDomainAxisLocation(AxisLocation location)
```

参数说明

location：表示 X 轴的显示位置。

设计过程

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("JAVA 图书", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集 CategoryDataset 实例，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "月份",                                           //X 轴标签
                        "销售量（单位：本）",                              //Y 轴标签
                        dataset,                                         //数据集
                        PlotOrientation.VERTICAL,                        //图表方向：水平、垂直
                        true,                                            //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                           //是否生成工具栏提示
                        false                                            //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中设置 X 轴显示在图表上方。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置 X 轴显示位置
        categoryPlot.setDomainAxisLocation(AxisLocation.TOP_OR_RIGHT);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

心法领悟 203：X 轴的显示位置。

使用 AxisLocation 类中的常量 AxisLocation.TOP_OR_RIGHT 可以为图表设置 X 轴显示位置，可以在图表的上方，也可以在图表右侧。在本实例步骤（3）中使用 PlotOrientation.VERTICAL 设置图表垂直显示时，X 轴将显示在图表上方；如果使用 PlotOrientation.HORIZONTAL 设置图表水平显示时，X 轴则显示在图表右侧。

8.6　Y 坐标轴



实例 204　Y 轴字体　高级

光盘位置：光盘\MR\08\204　实用指数：★★★

■ 实例说明

Y 轴是数据轴，一般情况下只用来显示数值，很少使用汉字。使用数值时，不会像使用汉字那样产生乱码。不过，也可以根据需要修改 Y 轴的字体。本实例运行结果如图 8.29 所示。

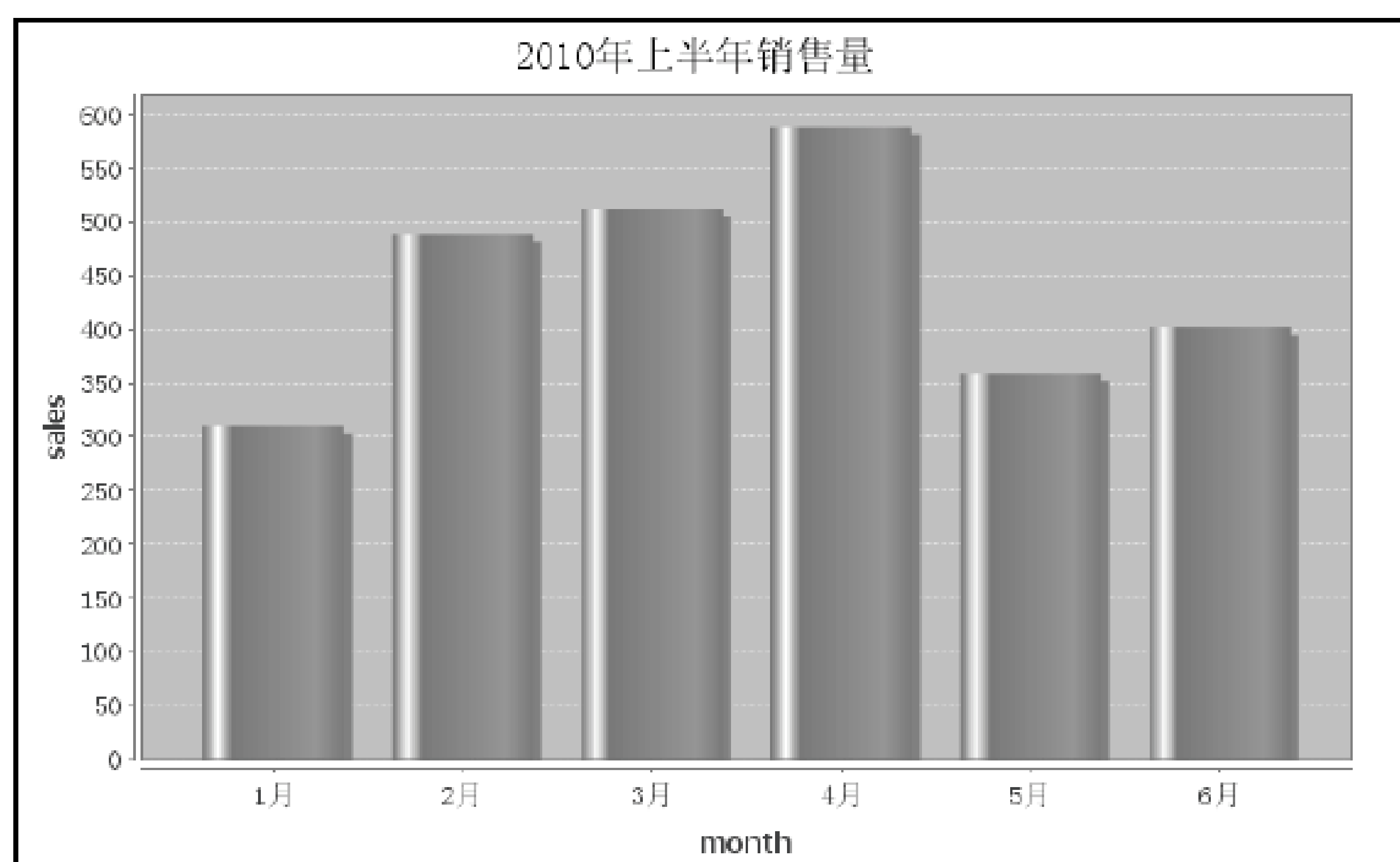


图 8.29　Y 轴字体

■ 关键技术

通过 JFreeChart 对象的 getCategoryPlot()方法可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 ValueAxis 对象，在 ValueAxis 对象中可以设置 Y 轴的字体。语法如下：

```
public void setTickLabelFont(Font font)
```

参数说明

font：表示 Y 轴的字体。

使用方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
ValueAxis valueAxis = categoryPlot.getRangeAxis();
//Y 轴字体
valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
```

■

（1）创建 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，然后通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010.年上半年销售量",   //图表标题
                        "month",                                          //X 轴标签
                        "sales",                                          //Y 轴标签
                        dataset,                                          //数据集
                        PlotOrientation.VERTICAL,                         //图表方向：水平、垂直
                        false,                                            //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                            //是否生成工具栏提示
                        false                                             //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改标题和 Y 轴的字体为“宋体”。代码如下：

```
public void updateFont(JFreeChart chart){
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        ValueAxis Valueaxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        Valueaxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 204：X、Y 轴的字体大小。

X 轴与 Y 轴是相辅相成的，一般情况下 X 轴的字体大小与 Y 轴的字体大小是一致的，所以当 X 轴字体大小设置为 14 时，Y 轴的字体大小应该也是 14。

实例 205　Y 轴标签字体

光盘位置：光盘\MR\08\205

高级　实用指数：★★★

实例说明

在柱形图中，X 轴又称分类轴，Y 轴又称数值轴，表示 Y 轴含义的文字被称为 Y 轴的标签。本实例将为 Y 轴的标签设置新的字体，使其能够支持汉字，如图 8.30 所示。

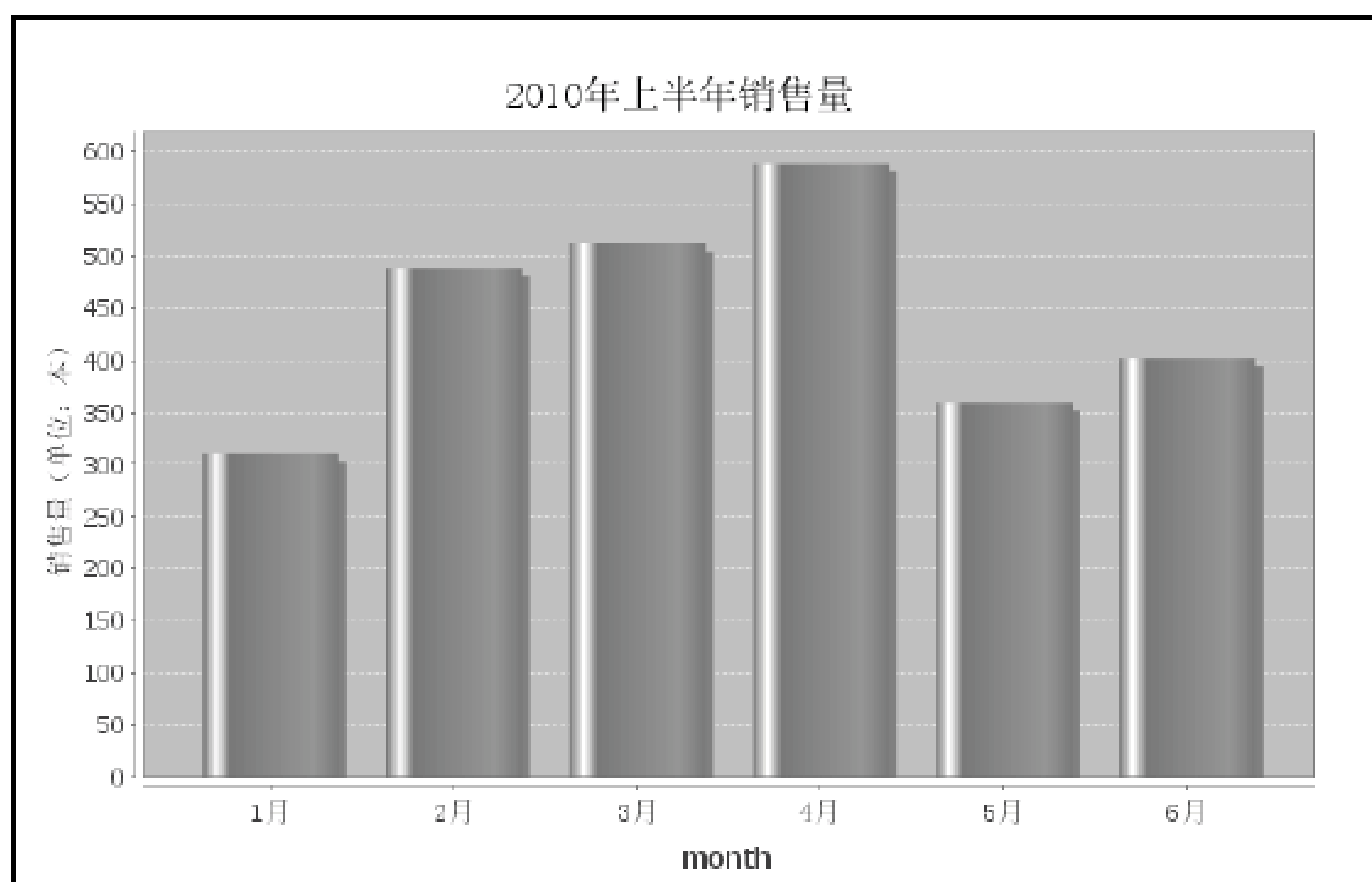


图 8.30　Y 轴标签字体

关键技术

通过 JFreeChart 对象的 getCategoryPlot()方法可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 ValueAxis 对象，在 ValueAxis 对象中可以设置 Y 轴标签的字体。语法如下：

```
public void setLabelFont(Font font)
```

参数说明

font：表示 Y 轴标签的字体。

使用方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
ValueAxis valueAxis = categoryPlot.getRangeAxis();
//Y 轴标签字体
valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
```

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "月份",                                          //X 轴标签
                        "销售量（单位：本）",                              //Y 轴标签
                        dataset,                                         //数据集
                        PlotOrientation.VERTICAL,                        //图表方向：水平、垂直
                        false,                                           //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                           //是否生成工具栏提示
                        false                                            //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改标题、X 轴、Y 轴、X 轴标签、Y 轴标签的字体为“宋体”。代码如下：

```
public static void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
```

```
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 205：Y 轴标签。

Y 轴是数值轴，其标签一般是用来表示 Y 轴的含义。可以在 Y 轴的标签后面为 Y 轴添加数值的单位，以更准确地表达图表的内容。

实例 206　Y 轴显示情况

光盘位置：光盘\MR\08\206

高级

实用指数：★★★

实例说明

在 JFreeChart 中柱形图的 Y 轴也可以被隐藏，如本实例就把 Y 轴的内容隐藏起来。Y 轴被隐藏起来以后，其相关设置也就没有任何意义了。本实例运行结果如图 8.31 所示。

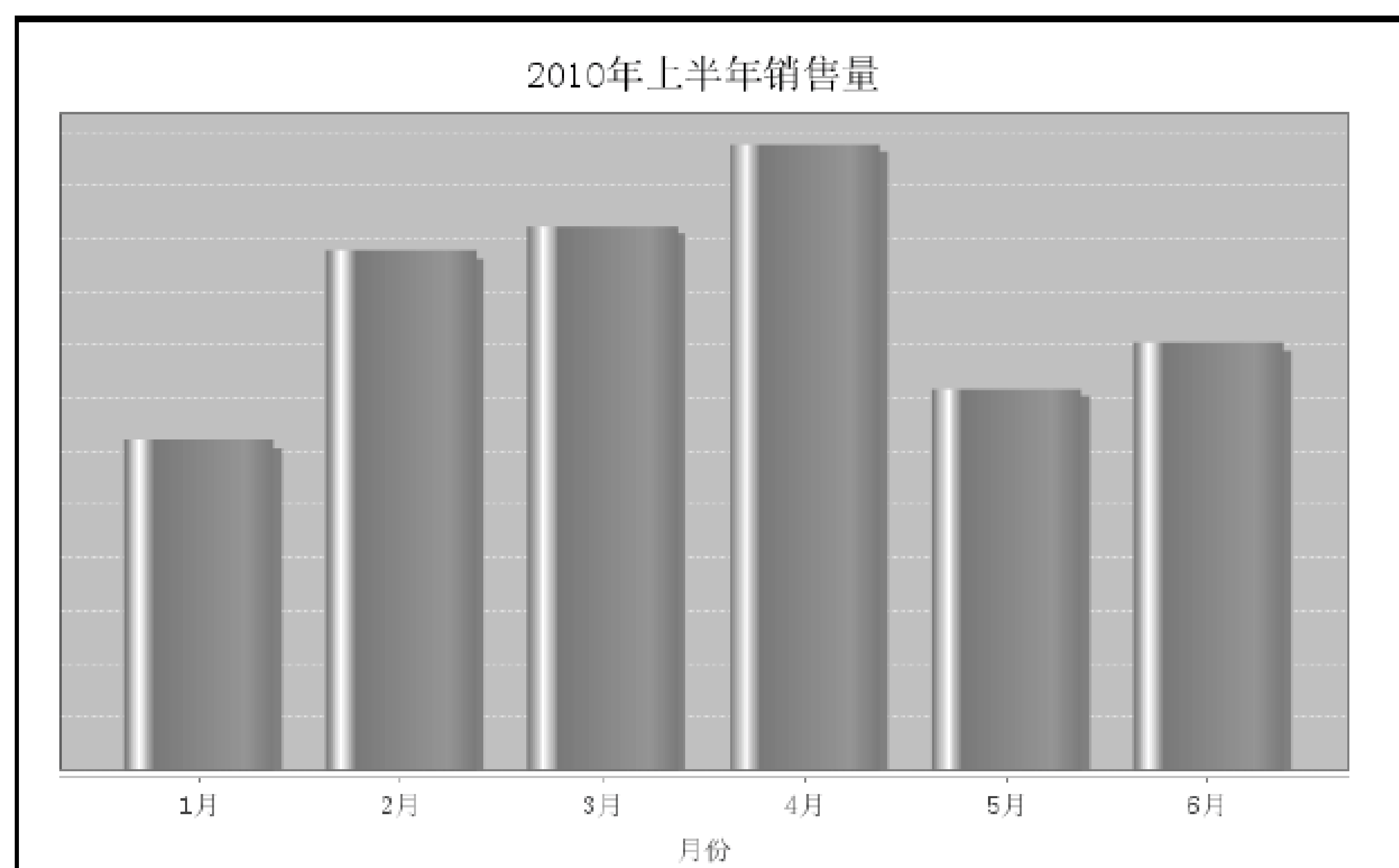


图 8.31　隐藏 Y 轴

通过 JFreeChart 对象的 getCategoryPlot()方法可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 ValueAxis 对象，在 ValueAxis 对象中使用 setVisible()方法可以隐藏 Y 轴。语法如下：

```
public void setVisible(boolean flag)
```

参数说明

flag：表示 Y 轴是否显示。当 flag 为 true 时显示 Y 轴；当 flag 为 false 时不显示 Y 轴。

使用方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
//Y 轴（数值轴）
ValueAxis valueAxis = categoryPlot.getRangeAxis();
//Y 轴是否显示
valueAxis.setVisible(false);
```

设计过程

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                "月份",                                      //X 轴标签
                "销售量（单位：本）",                        //Y 轴标签
                dataset,                                     //数据集
                PlotOrientation.VERTICAL,                    //图表方向：水平、垂直
                false,                                       //是否显示图例（对于简单的柱形图必须是 false）
                false,                                       //是否生成工具栏提示
                false                                        //是否生成 URL 链接
                );
        return chart;
}
```

（3）创建 updatePlot()方法，获取 ValueAxis 的实例，使用 ValueAxis 的实例隐藏 Y 轴。代码如下：

```
public static void updatePlot(JFreeChart chart){
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //Y 轴（数值轴）
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴是否显示
        valueAxis.setVisible(false);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 206：Y 轴的隐藏。

JFreeChart 在隐藏 Y 轴时，会同时隐藏 Y 轴的尺度线、尺度文字以及 Y 轴标签，所以在没有特殊要求的情况下 Y 轴是不需要隐藏的，Y 轴的默认设置也是处于显示的状态。

实例 207	Y 轴尺度线颜色和笔触 光盘位置：光盘\MR\08\207	高级 实用指数：★★

实例说明

在 JFreeChart 中，柱形图中的 Y 轴尺度线的颜色应该和 X 轴尺度线颜色一致，本实例将把 Y 轴的尺度线改成和 X 轴同样的绿色。此外，Y 轴的尺度线可以根据实际情况进行调整，例如将其加粗、修改其样式，本实例

将把 Y 轴的尺度线加粗到 5。实例运行结果如图 8.32 所示。

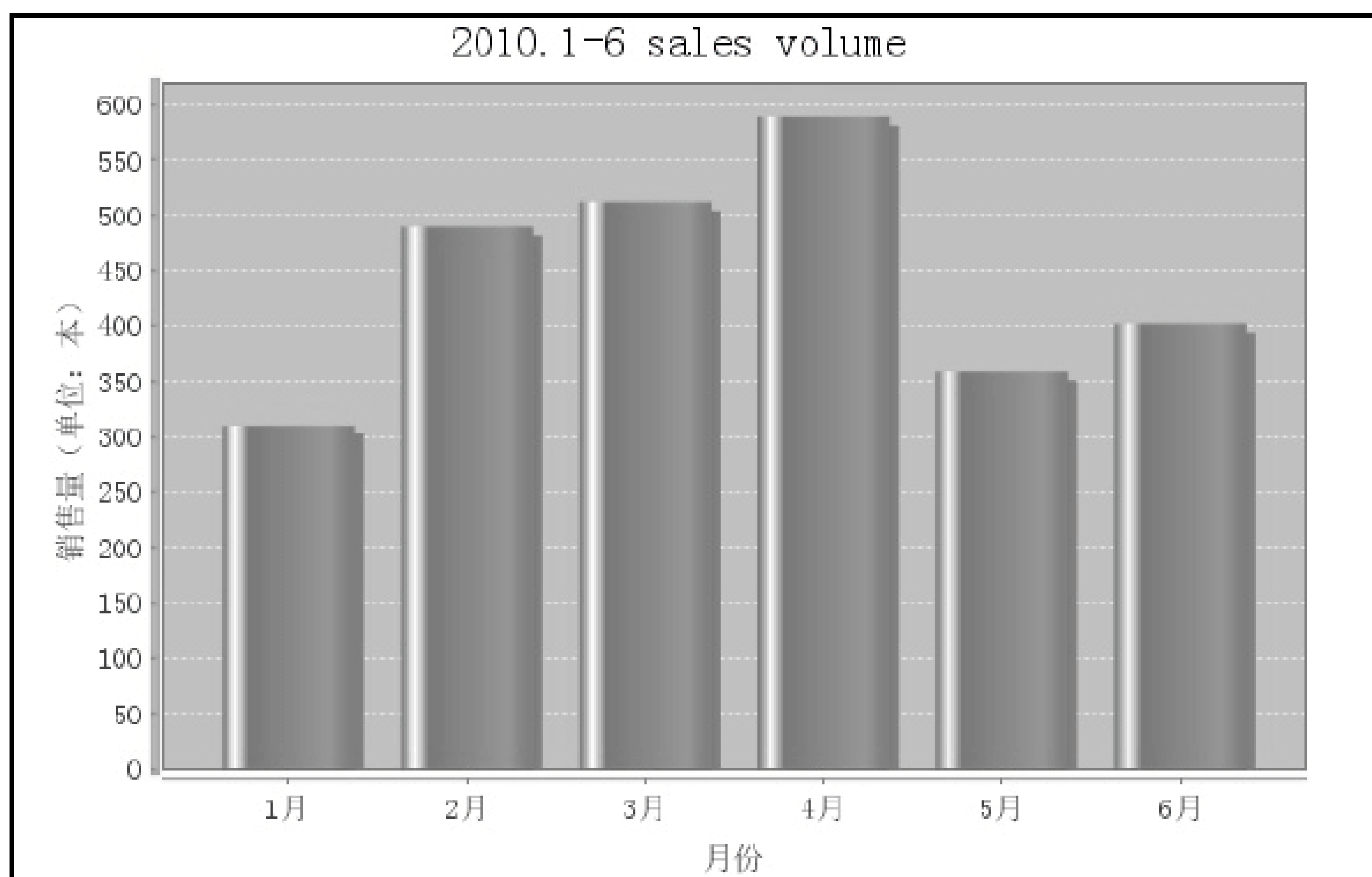


图 8.32　Y 轴尺度线颜色和笔触

关键技术

（1）通过 JFreeChart 对象的 getCategoryPlot()方法可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 ValueAxis 对象，在 ValueAxis 对象中可以为 Y 轴尺度线设置颜色。语法如下：

```
public void setAxisLinePaint(Paint paint)
```

参数说明

paint：表示 Y 轴尺度线设置的颜色。

使用方法如下：

```
CategoryPlot categoryPlot = chart.getCategoryPlot();    //图表（柱形图）
ValueAxis valueAxis = categoryPlot.getRangeAxis();    //Y 轴（数值轴）
valueAxis.setAxisLinePaint(Color.GREEN);              //Y 轴尺度颜色
```

（2）通过 JFreeChart 对象的 getCategoryPlot()方法可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 ValueAxis 对象，在 ValueAxis 对象中可以使用如下方法加粗 Y 轴尺度线。

```
public BasicStroke(float width)
```

参数说明

width：表示笔触要加粗的数值。

使用方法如下：

```
CategoryPlot categoryPlot = chart.getCategoryPlot();    //图表（柱形图）
ValueAxis valueAxis = categoryPlot.getRangeAxis();    //Y 轴
valueAxis.setAxisLineStroke(new BasicStroke(5));      //尺度线笔触
```


（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。

代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010.1-6 sales volume",    //图表标题
                        "月份",                                                     //X 轴标签
                        "销售量（单位：本）",                                        //Y 轴标签
                        dataset,                                                   //数据集
                        PlotOrientation.VERTICAL,                                  //图表方向：水平、垂直
                        false,                                                     //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                                     //是否生成工具栏提示
                        false                                                      //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updatePlot()方法，将 X、Y 轴尺度线颜色设置为绿色，并且将 Y 轴尺度线加粗为 5。代码如下：

```
public static void updatePlot(JFreeChart chart){
        CategoryPlot categoryPlot = chart.getCategoryPlot();            //图表（柱形图）
        ValueAxis valueAxis = categoryPlot.getRangeAxis();              //Y 轴（数值轴）
        valueAxis.setAxisLinePaint(Color.GREEN);                        //Y 轴尺度线颜色
        valueAxis.setAxisLineStroke(new BasicStroke(5));                //尺度线笔触
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();       //X 轴（分类轴）
        categoryAxis.setAxisLinePaint(Color.GREEN);                     //X 轴尺度线颜色
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 207：设置 Y 轴尺度线颜色。

在 JFreeChart 的图表中，Y 轴尺度线的颜色默认为灰色，在 Java 的 Color 类中的常量为 Color.GRAY。本实例设置 Y 轴尺度线为绿色时，使用了 Color 常量 Color.GREEN。如果希望修改 Y 轴字体的颜色，可以使用 ValueAxis 的 setTickLabelPaint()方法。

实例 208	隐藏 Y 轴尺度线 光盘位置：光盘\MR\08\208	高级 实用指数：★★

实例说明

Y 轴的尺度线在图表的 Y 轴左侧多出一条有刻度的线，在使用时可以根据需要将其去掉。本实例将演示如何去掉 Y 轴的尺度线，运行结果如图 8.33 所示。

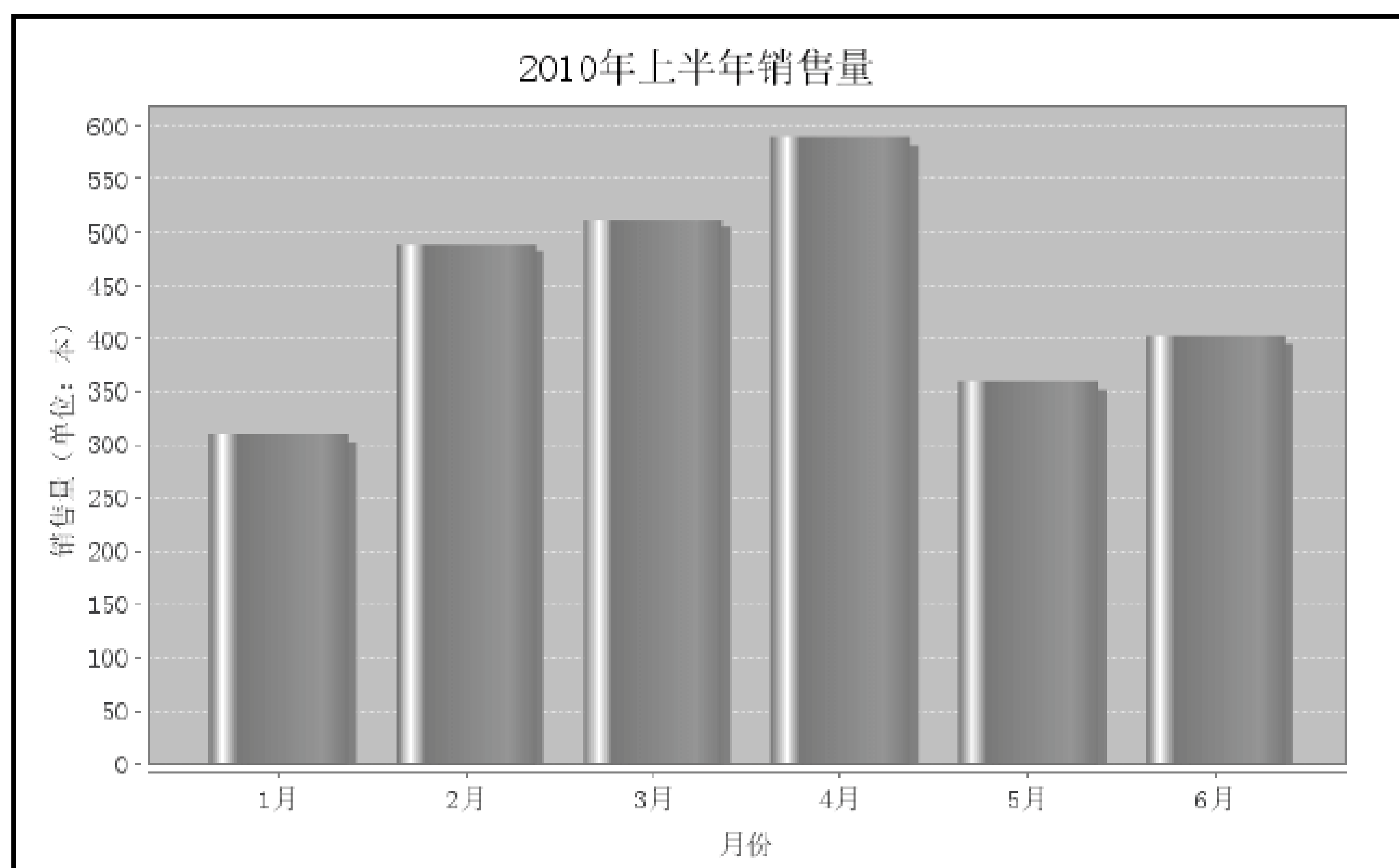


图 8.33　隐藏 Y 轴尺度线

关键技术

通过 JFreeChart 对象的 getCategoryPlot()方法可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 ValueAxis 对象，在 ValueAxis 对象中可以隐藏 Y 轴尺度线。语法如下：

```
public void setAxisLineVisible(boolean visible)
```

参数说明

visible：表示 Y 轴尺度线是否显示。当 visible 为 true 时，显示尺度线；当 visible 为 false 时，则不显示尺度线。

使用方法如下：

```
        CategoryPlot categoryPlot = chart.getCategoryPlot();          //图表（柱形图）
        ValueAxis valueAxis = categoryPlot.getRangeAxis();            //Y 轴
        valueAxis.setAxisLineVisible(false);                          //尺度线
```

设计过程

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "月份",                                               //X 轴标签
                        "销售量（单位：本）",                                   //Y 轴标签
                        dataset,                                              //数据集
                        PlotOrientation.VERTICAL,                             //图表方向：水平、垂直
                        false,                                                //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                                //是否生成工具栏提示
                        false                                                 //是否生成 URL 链接
                        );
        return chart;
}
```

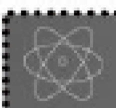
（3）创建 updatePlot()方法，在该方法中隐藏 Y 轴尺度线。代码如下：

```
public static void updatePlot(JFreeChart chart){
        CategoryPlot categoryPlot = chart.getCategoryPlot();          //图表（柱形图）
        ValueAxis valueAxis = categoryPlot.getRangeAxis();            //Y 轴
        valueAxis.setAxisLineVisible(false);                          //尺度线
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

心法领悟 208：Y 轴的尺度线。

Y 轴的尺度线在默认情况下是显示的，使用者可以根据需要决定是否使用尺度线。如果 Y 轴的尺度线被隐藏，尺度线上的刻度会自动显示在图表的左边框上。



实例 209　Y 轴尺度标签角度

光盘位置：光盘\MR\08\209

高级

实用指数：★★★

实例说明

Y 轴尺度的标签角度可以根据使用者的需要进行调整，本实例将把 Y 轴尺度的标签逆时针旋转 90°，运行结果如图 8.34 所示。

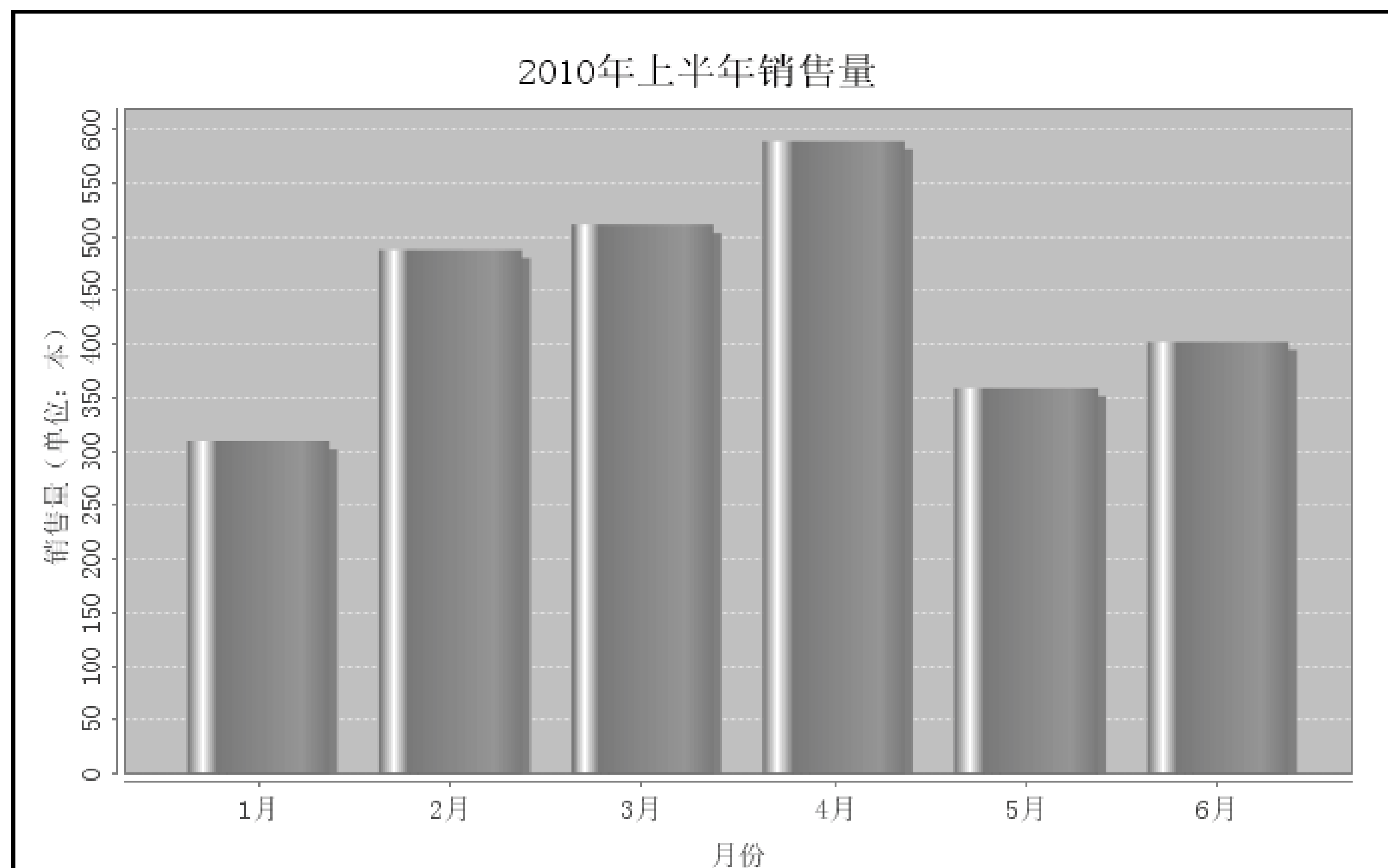


图 8.34　Y 轴尺度标签角度

关键技术

通过 JFreeChart 对象的 getCategoryPlot()方法可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 ValueAxis 对象，在 ValueAxis 对象中使用如下方法可以修改 Y 轴尺度标签的角度。

```
public void setVerticalTickLabels(boolean flag)
```

参数说明

flag：表示 Y 轴尺度标签是否垂直。当 flag 为 true 时，表示 Y 轴尺度标签是垂直的；当 flag 为 false 时，表示 Y 轴尺度标签不是垂直的，即水平的。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
	CategoryDataset dataset = getCategoryDataset();
	JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",	//图表标题
			"月份",				//X 轴标签
			"销售量（单位：本）",		//Y 轴标签
			dataset,				//数据集
			PlotOrientation.VERTICAL,	//图表方向：水平、垂直
			false,				//是否显示图例（对于简单的柱形图必须是 false）
			false,				//是否生成工具栏提示
			false				//是否生成 URL 链接
			);
	return chart;
}
```

（3）创建 updatePlot()方法，在该方法中设置 Y 轴尺度标签垂直。代码如下：

```
public static void updatePlot(JFreeChart chart){
	CategoryPlot categoryPlot = chart.getCategoryPlot();		//图表
	ValueAxis valueAxis = categoryPlot.getRangeAxis();		//Y 轴
	valueAxis.setVerticalTickLabels(true);			//Y 轴尺度标签
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 209：Y 轴尺度标签的角度与 X 轴尺度标签的角度的区别。

Y 轴尺度标签的角度与 X 轴不同，X 轴尺度标签中 JFreeChart 提供了 5 个角度，分别使用不同的常量来表示；而 Y 轴尺度标签的角度控制中，JFreeChart 只提供了一个 Boolean 类型的方法。

实例 210 Y 轴起始值

光盘位置：光盘\MR\08\210

高级

实用指数：★★★

实例说明

一般情况下，柱形图的起始值默认为 0，本实例修改了柱形图中 Y 轴的起始值，让图表有更多的区域展示不同柱形之间的区别，运行结果如图 8.35 所示。

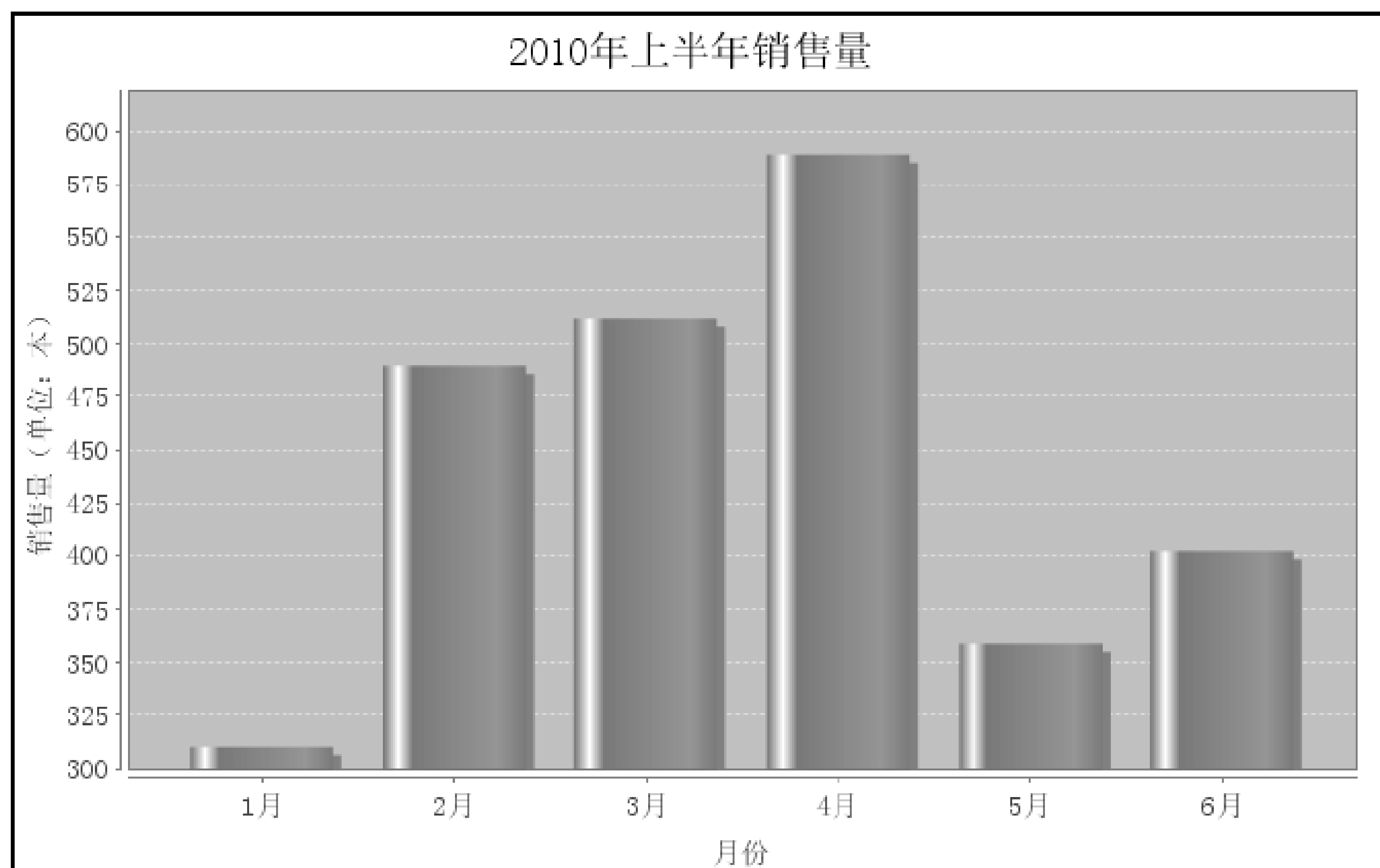


图 8.35　设置柱形图起始值

关键技术

设置柱形图的起始值主要用到 ValueAxis 类的 setLowerBound()方法，该方法可以指定柱形图 Y 轴的起始值。语法如下：

```
public void setLowerBound(double min)
```

参数说明

min：表示 Y 轴的起始值。

设计过程

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "月份",                                          //X 轴标签
                        "销售量（单位：本）",                              //Y 轴标签
                        dataset,                                         //数据集
                        PlotOrientation.VERTICAL,                        //图表方向：水平、垂直
                        false,                                           //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                           //是否生成工具栏提示
                        false                                            //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 ValueAxis 的对象，并且设置 Y 轴的起始值为 300。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        CategoryPlot categoryPlot = chart.getCategoryPlot();           //图表
        ValueAxis valueAxis = categoryPlot.getRangeAxis();             //Y 轴
        valueAxis.setLowerBound(300);                                  //Y 轴起始值
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 210：Y 轴的起始值。

设置 Y 轴的起始值，主要是为了让柱形图从 Y 轴的起始值开始显示图形。一般情况下，设置的 Y 轴的起始值都不会大于最低的柱形的高度。

实例 211	Y 轴箭头 光盘位置：光盘\MR\08\211	高级 实用指数：★★

在柱形图 Y 轴的上端和下端都可以添加箭头来表示数据延伸的方向。本实例为柱形图 Y 轴添加了上端箭头，运行结果如图 8.36 所示。

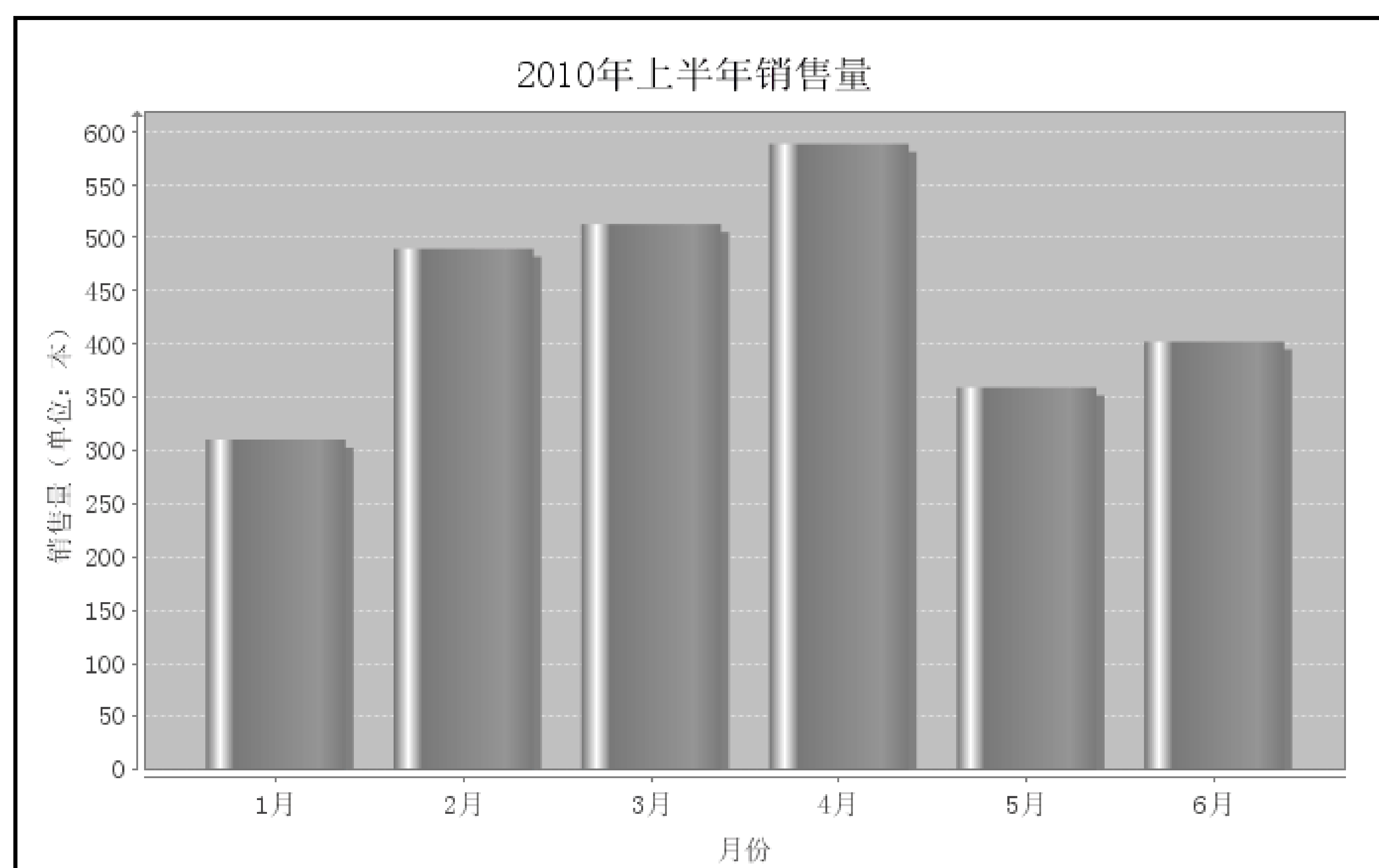


图 8.36　设置 Y 轴上端的箭头

关键技术

设置柱形图 Y 轴上端的箭头主要用到 ValueAxis 类的 setPositiveArrowVisible()方法，该方法通过 true 和 false 表示是否显示 Y 轴上端的箭头。

语法如下：

```
public void setPositiveArrowVisible(boolean visible)
```

参数说明

visible：表示 Y 轴上端的箭头是否显示。当 visible 为 true 时，显示 Y 轴上端的箭头；当 visible 为 false 时，不显示 Y 轴上端的箭头。默认值为 false。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                "月份",                                                    //X 轴标签
                "销售量（单位：本）",                                        //Y 轴标签
                dataset,                                                   //数据集
                PlotOrientation.VERTICAL,                                  //图表方向：水平、垂直
                false,                                                     //是否显示图例（对于简单的柱形图必须是 false）
                false,                                                     //是否生成工具栏提示
                false                                                      //是否生成 URL 链接
                );
```

```
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 ValueAxis 的对象，并且显示出 Y 轴上端的箭头。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        CategoryPlot categoryPlot = chart.getCategoryPlot();    //图表
        ValueAxis valueAxis = categoryPlot.getRangeAxis();     //Y 轴
        valueAxis.setPositiveArrowVisible(true);               //是否显示 Y 轴向上箭头
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 211：利用 setNegativeArrowVisible()方法设置下端的箭头。

使用 setPositiveArrowVisible()方法可以设置是否显示 Y 轴上端的箭头，而使用 setNegativeArrowVisible()方法还可以设置下端的箭头是否显示。语法如下：

```
setNegativeArrowVisible(boolean visible)
```

参数说明

visible：表示 Y 轴下端的箭头是否显示。当 visible 为 true 时，显示 Y 轴下端的箭头；当 visible 为 false 时，不显示 Y 轴下端的箭头。默认值为 false。

实例 212	隐藏 Y 轴主要刻度线 光盘位置：光盘\MR\08\212	高级 实用指数：★★★

实例说明

通常所说的刻度线指的是主要刻度线。柱形图上 Y 轴的主要刻度线默认处于显示状态，也可根据实际需要将其隐藏。本实例运行结果如图 8.37 所示。

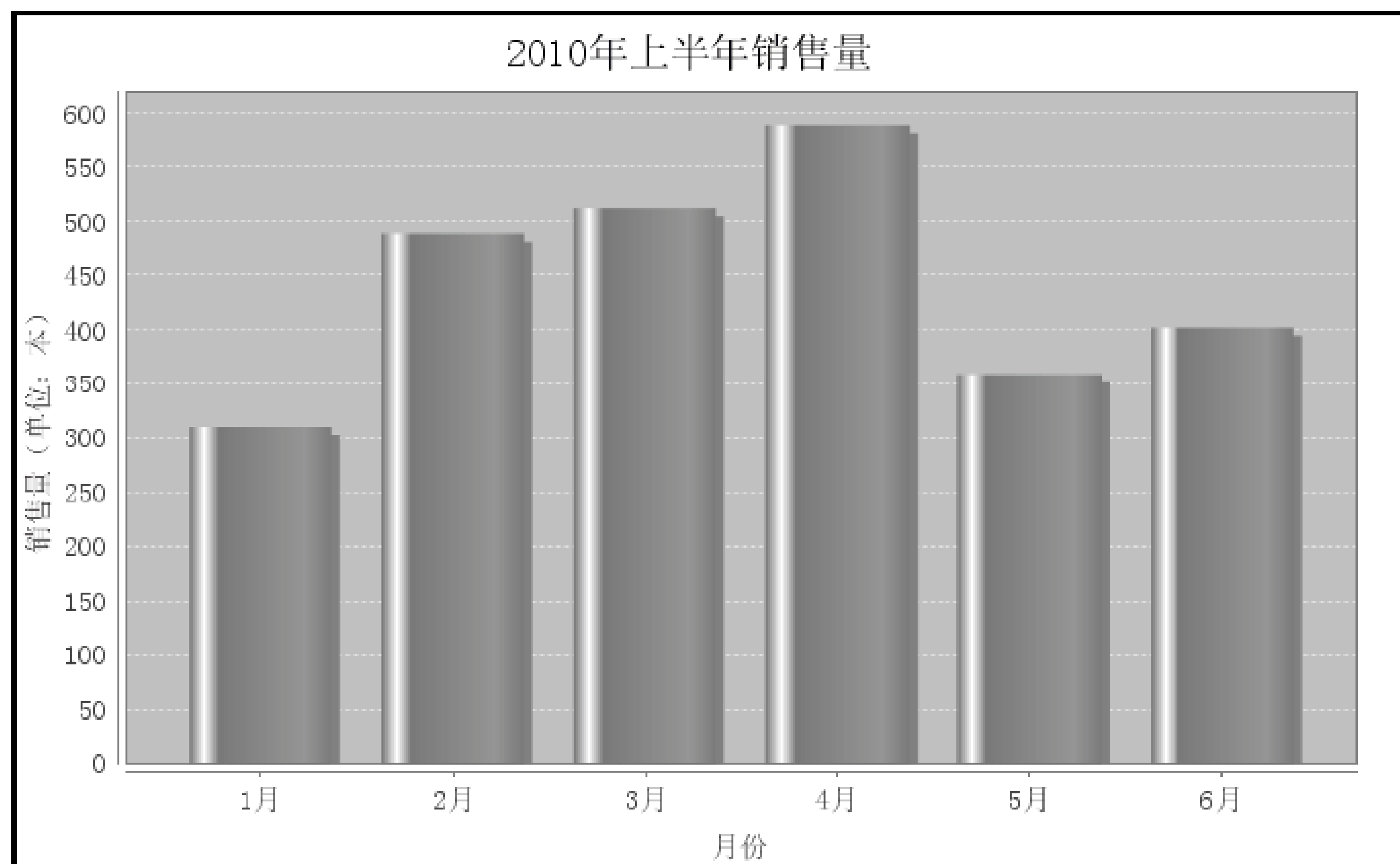


图 8.37　隐藏 Y 轴主要刻度线

柱形图的 Y 轴刻度线可以根据需要显示或者隐藏，在 JFreeChart 中使用 Axis 类的 setTickMarksVisible()方法进行控制。

语法如下：

```
public void setTickMarksVisible (boolean flag)
```

参数说明

flag：表示 Y 轴上是否显示主要刻度线。当 flag 为 true 时，显示刻度线；当 flag 为 false 时，不显示刻度线。

设计过程

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "月份",                                            //X 轴标签
                        "销售量（单位：本）",                                //Y 轴标签
                        dataset,                                           //数据集
                        PlotOrientation.VERTICAL,                          //图表方向：水平、垂直
                        false,                                             //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                             //是否生成工具栏提示
                        false                                              //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 ValueAxis 的对象，并且隐藏 Y 轴的主要刻度线。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        CategoryPlot categoryPlot = chart.getCategoryPlot();             //图表
        ValueAxis valueAxis = categoryPlot.getRangeAxis();               //Y 轴
        valueAxis.setTickMarksVisible(false);                            //刻度线是否显示
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 212：Y 轴的主要刻度线。

Y 轴的主要刻度线一般情况下都是显示的。柱形图的背景中有一些白色虚线，它们就是以 Y 轴的主要刻度线分布的，即使隐藏了刻度线，这些白色虚线也会正常显示。

实例 213	Y 轴主要刻度线长度 光盘位置：光盘\MR\08\213	高级 实用指数：★★★

实例说明

Y 轴的主要刻度线默认情况下只有外部刻度线，但实际上无论是在图形内部，还是图形外部主要刻度线都可以设置长度。本实例将显示出主要刻度，并且设置其长度，运行结果如图 8.38 所示。

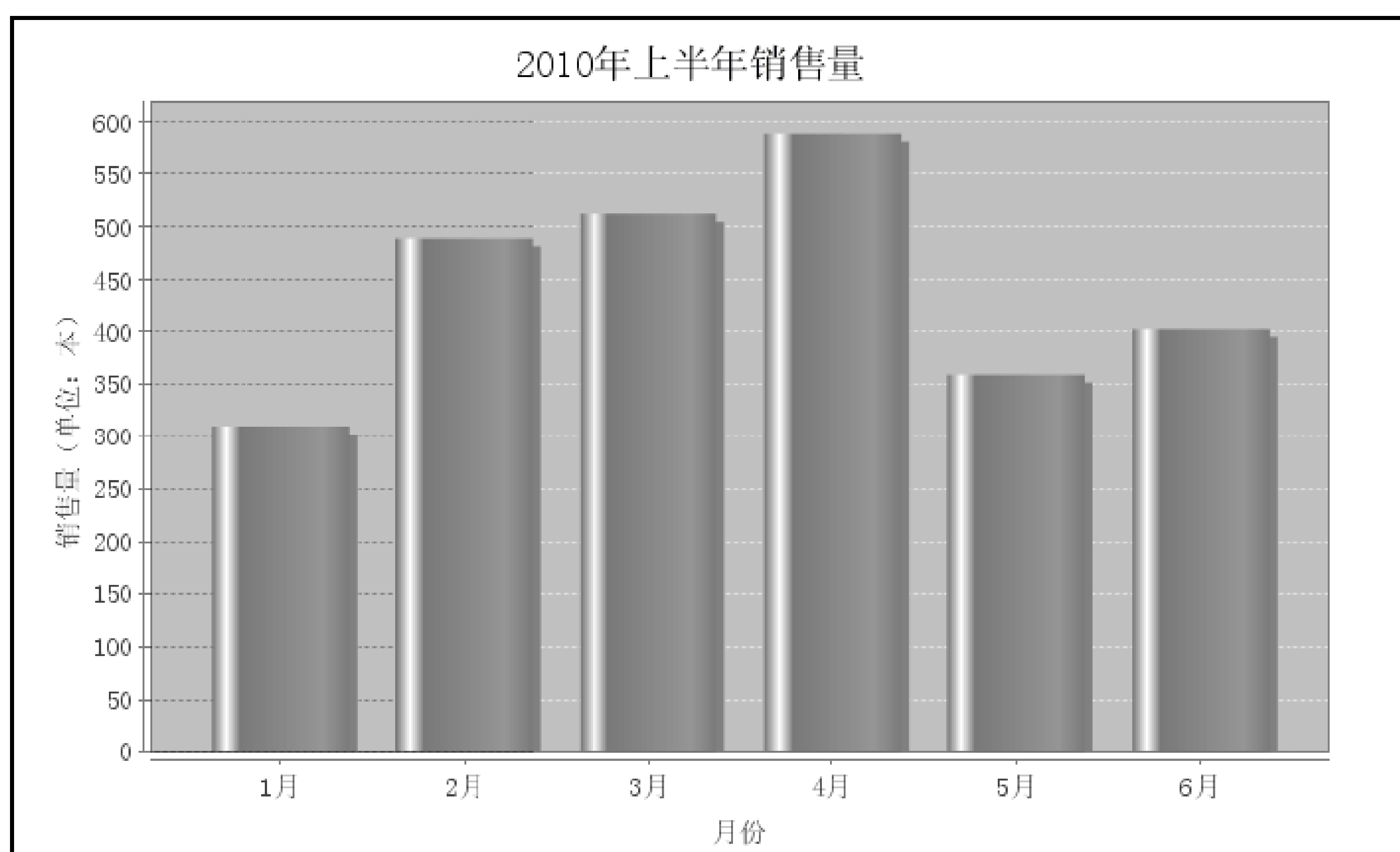


图 8.38　设置 Y 轴主要刻度线长度

关键技术

（1）设置柱形图 Y 轴内部主要刻度线长度主要用到 Axis 类的 setTickMarkInsideLength()方法，通过该方法可以根据主要刻度线在柱形图内部显示线条。

语法如下：

```
public void setTickMarkInsideLength (float length)
```

参数说明

length：表示 Y 轴主要刻度线内部延长线的长度。

（2）通过 Axis 类的 setTickMarkOutsideLength()方法还可以设置 Y 轴主要刻度线外部的延长线长度。

语法如下：

```
public void setTickMarkOutsideLength (float length)
```

length：表示 Y 轴主要刻度线外部延长线的长度。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "月份",                                          //X 轴标签
                        "销售量（单位：本）",                              //Y 轴标签
                        dataset,                                        //数据集
                        PlotOrientation.VERTICAL,                       //图表方向：水平、垂直
```

```
                    false,                                    //是否显示图例（对于简单的柱形图必须是 false）
                    false,                                    //是否生成工具栏提示
                    false                                     //是否生成 URL 链接
                    );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 ValueAxis 的对象，然后将 Y 轴主要刻度线的内部延长线长度设置为 200。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        CategoryPlot categoryPlot = chart.getCategoryPlot();          //图表
        ValueAxis valueAxis = categoryPlot.getRangeAxis();            //Y 轴
        valueAxis.setTickMarksVisible(true);                          //刻度线是否显示
        valueAxis.setTickMarkInsideLength(200);                       //内部刻度线
        //valueAxis.setTickMarkOutsideLength(20);                     //外部刻度线
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 213：Y 轴主要刻度线。

默认情况下，Y 轴主要刻度线处于显示状态。它由两部分组成，一部分是外部主要刻度线，也就是 Y 轴左侧的线，默认情况下显示的就是这部分的刻度线，长度为 2；另一部分是 Y 轴右侧的线，是内部主要刻度线，其默认值为 0，一般情况下不显示。

实例 214 设置 Y 轴最大值

光盘位置：光盘\MR\08\214

高级

实用指数：★★★

实例说明

在 JFreeChart 中，Y 轴的最大值一般情况下是 JFreeChart 根据数据集中的数据动态设置的。本实例中屏蔽了 JFreeChart 自动设置的最大值，进行人工设置，运行结果如图 8.39 所示。

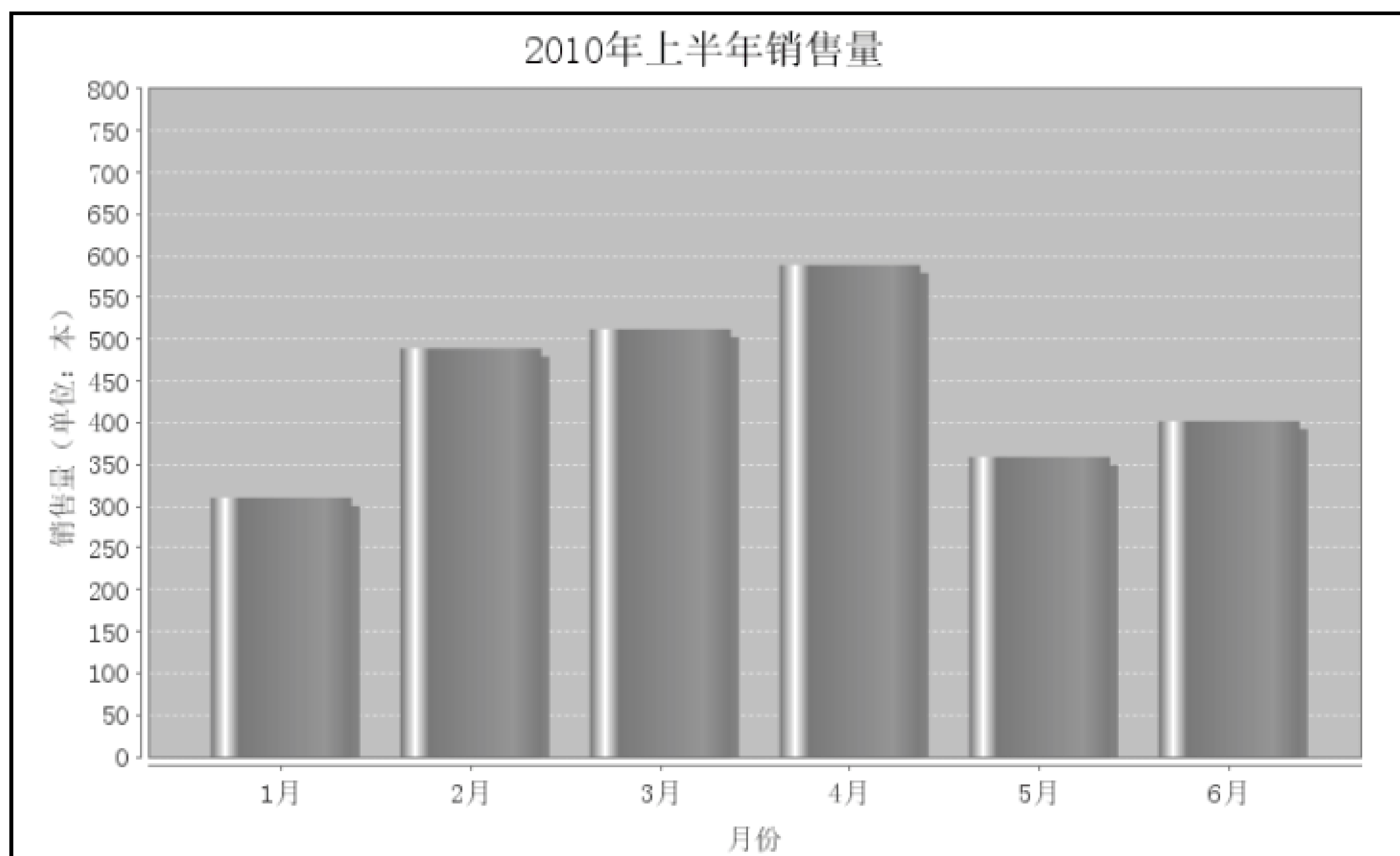


图 8.39　设置 Y 轴最大值

关键技术

如果已经确定了数据的最大值，则可以使用 ValueAxis 类的 setupperBound()方法对 Y 轴的最大值进行人工设置。

语法如下：

```
public void setUpperBound(double max)
```

参数说明

max：表示 Y 轴的最大值。

设计过程

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",    //图表标题
                        "月份",                                           //X 轴标签
                        "销售量（单位：本）",                               //Y 轴标签
                        dataset,                                          //数据集
                        PlotOrientation.VERTICAL,                         //图表方向：水平、垂直
                        false,                                            //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                            //是否生成工具栏提示
                        false                                             //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 ValueAxis 的对象，然后为 Y 轴设置最大值为 800。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        CategoryPlot categoryPlot = chart.getCategoryPlot();          //图表
        ValueAxis valueAxis = categoryPlot.getRangeAxis();            //Y 轴
        valueAxis.setUpperBound(800);                                 //Y 轴最大值
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 214：设置 Y 轴最大值。

设置 Y 轴最大值时，最大值一定要超过柱形图中数据集里最大的值，否则在 JFreeChart 生成图像以后，柱形的图像会占满整个区域，无法分辨柱形图的高度。

实例 215	设置 Y 轴数据范围 光盘位置：光盘\MR\08\215	高级 实用指数：★★★

实例说明

Y 轴的数据范围默认情况下也是由 JFreeChart 设置的，本实例演示如何人工地为 Y 轴设置数据范围，运行

效果如图 8.40 所示。

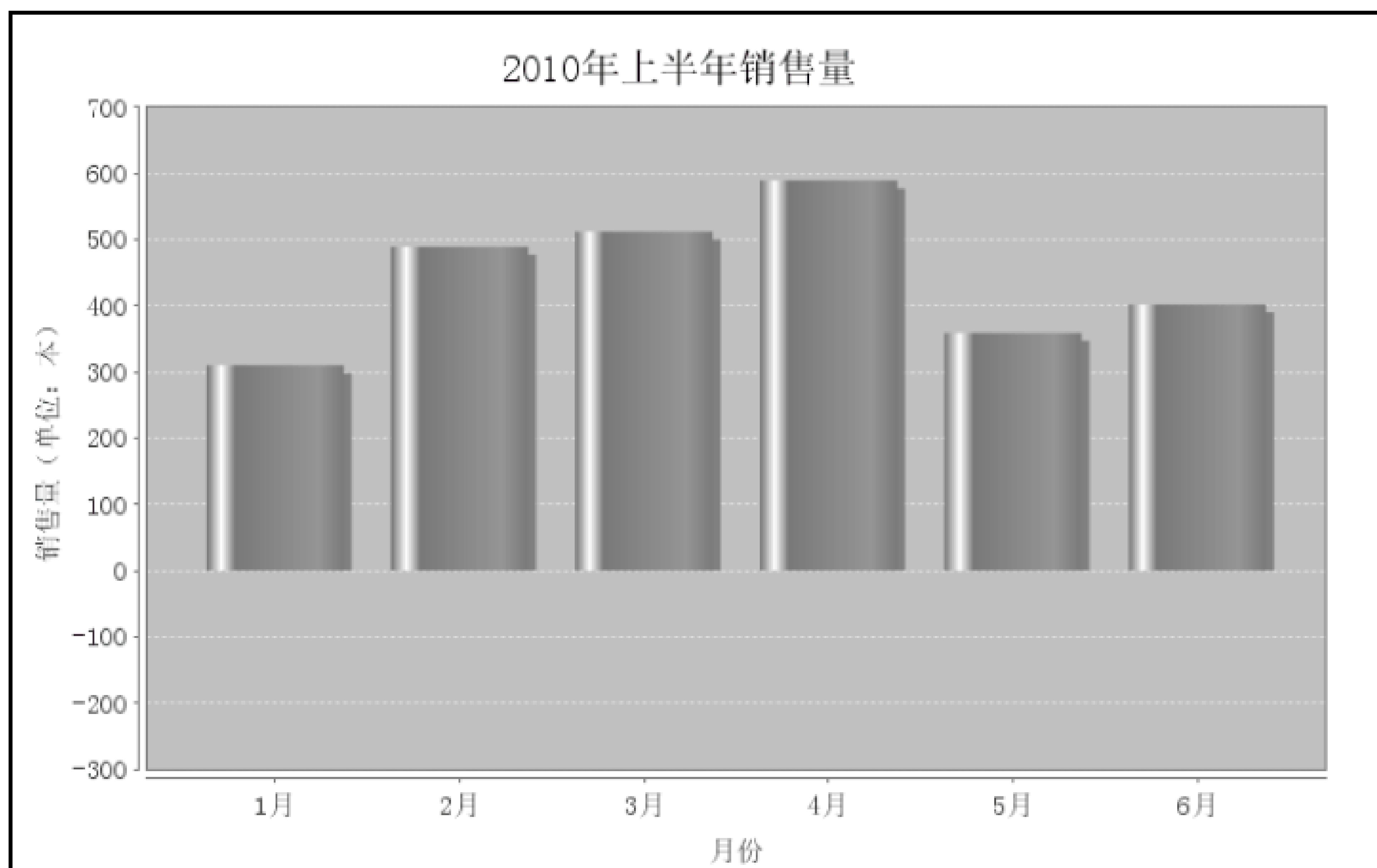


图 8.40　设置 Y 轴数据范围

关键技术

使用 ValueAxis 类的 setRangeAboutValue()方法可以设置 Y 轴的数据范围。语法如下：

```
public void setRangeAboutValue(double value, double length)
```

参数说明

❶ value：表示 Y 轴的中间值。

❷ length：表示 Y 轴的总数值。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "月份",                                              //X 轴标签
                        "销售量（单位：本）",                                //Y 轴标签
                        dataset,                                             //数据集
                        PlotOrientation.VERTICAL,                            //图表方向：水平、垂直
                        false,                                               //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                               //是否生成工具栏提示
                        false                                                //是否生成 URL 链接
```

```
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 ValueAxis 的对象，通过设置 Y 轴的中间值和总数值确定其数据范围。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        CategoryPlot categoryPlot = chart.getCategoryPlot();    //图表
        ValueAxis valueAxis = categoryPlot.getRangeAxis();      //Y 轴
        valueAxis.setRangeAboutValue(200, 1000);                //Y 轴数据范围
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 215：为 Y 轴设置总数值。

为 Y 轴设置总数值时要注意，总数值必须大于数据集中最大值与最小值的绝对值之和；而设置中间值时，一般取总数值的一半即可，否则柱形图像显示得不完整。

实例 216	Y 轴的显示位置 光盘位置：光盘\MR\08\216	高级 实用指数：★★★★

实例说明

一般的柱形图都是垂直显示，Y 轴一般都在图表的左侧，但有时根据需要也可能会对 Y 轴进行调整。本实例将演示如何把 Y 轴显示在图表右侧，运行结果如图 8.41 所示。

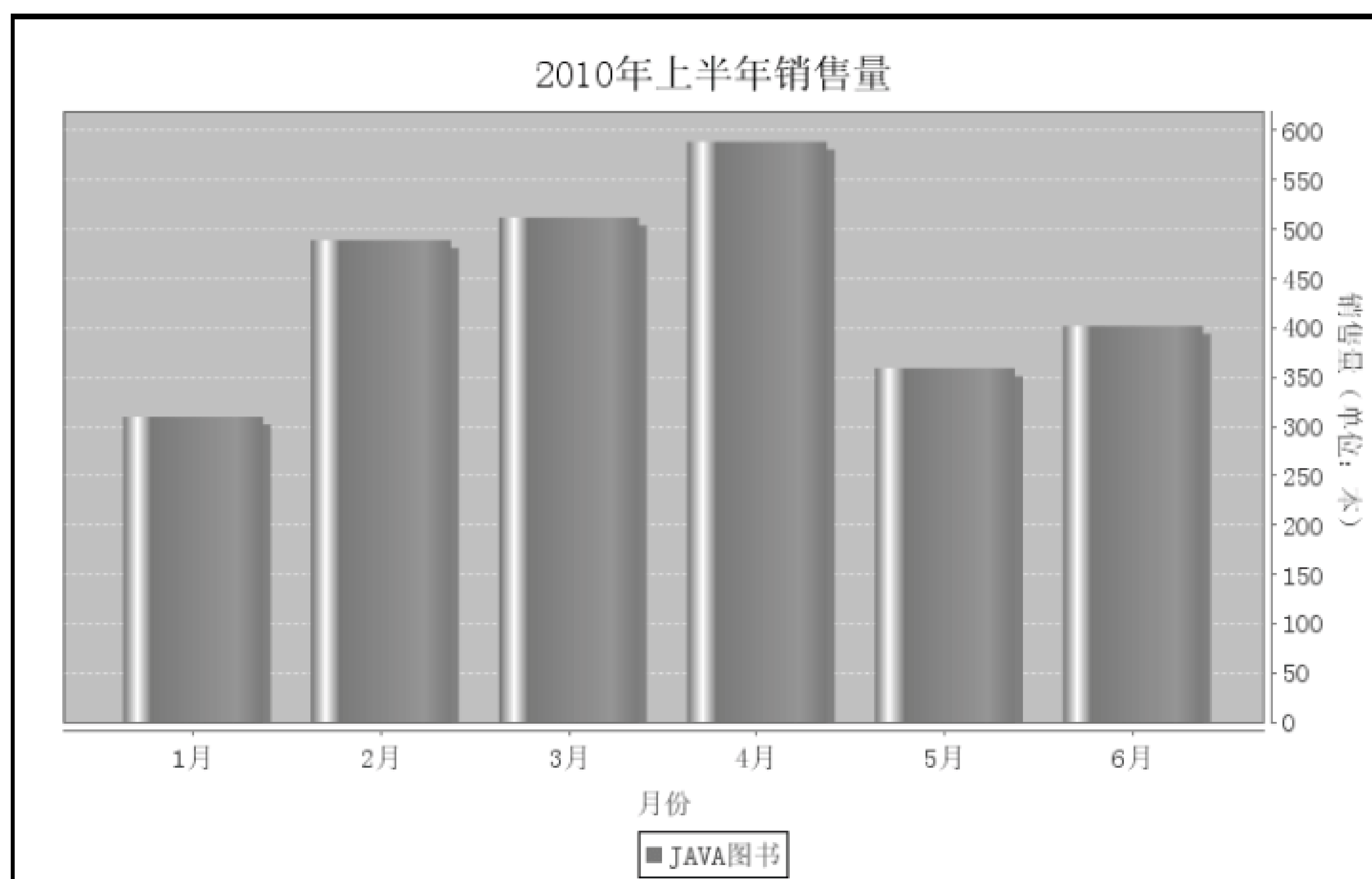


图 8.41　设置 Y 轴显示位置

关键技术

使用 CategoryPlot 类的 setRangeAxisLocation()方法可以设置 Y 轴在图表中的显示位置。语法如下：

```
Public void setRangeAxisLocation(AxisLocation location)
```

参数说明

location：表示 Y 轴的显示位置。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。

代码如下：

```
private static CategoryDataset getCategoryDataset() {
    DefaultKeyedValues keyedValues = new DefaultKeyedValues();
    //添加数据
    keyedValues.addValue("1 月", 310);
    keyedValues.addValue("2 月", 489);
    keyedValues.addValue("3 月", 512);
    keyedValues.addValue("4 月", 589);
    keyedValues.addValue("5 月", 359);
    keyedValues.addValue("6 月", 402);
    //创建数据集合实例
    CategoryDataset dataset = DatasetUtilities.createCategoryDataset("JAVA 图书", keyedValues);
    return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集 CategoryDataset 实例，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
    CategoryDataset dataset = getCategoryDataset();
    JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
            "月份",                                                //X 轴标签
            "销售量（单位：本）",                                    //Y 轴标签
            dataset,                                              //数据集
            PlotOrientation.VERTICAL,                             //图表方向：水平、垂直
            true,                                                 //是否显示图例（对于简单的柱形图必须是 false）
            false,                                                //是否生成工具栏提示
            false                                                 //是否生成 URL 链接
            );
    return chart;
}
```

（3）创建 updatePlot()方法，在该方法中设置 Y 轴显示在图表右侧。代码如下：

```
public static void updatePlot(JFreeChart chart) {
    CategoryPlot categoryPlot = chart.getCategoryPlot();                 //图表
    categoryPlot.setRangeAxisLocation(AxisLocation.TOP_OR_RIGHT);        //设置 Y 轴显示位置
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 216：使用 AxisLocation 类中的常量设置 Y 轴显示位置。

使用 AxisLocation 类中的常量 AxisLocation.TOP_OR_RIGHT 可以为图表设置 Y 轴显示位置，可以在图表的上方，也可以在图表右侧。在本实例步骤（3）中使用 PlotOrientation.VERTICAL 设置图表垂直显示时，Y 轴将显示在图表右侧；如果使用 PlotOrientation.HORIZONTAL 设置图表水平显示时，Y 轴则显示在图表上方。

8.7 高级柱形图

实例 217	设置网格竖线 光盘位置：光盘\MR\08\217	高级 实用指数：★★★

实例说明

默认情况下，柱形图图像区中只显示网格横线，网格竖线一般是不显示的。本实例将演示如何显示网格竖线，运行结果如图 8.42 所示。

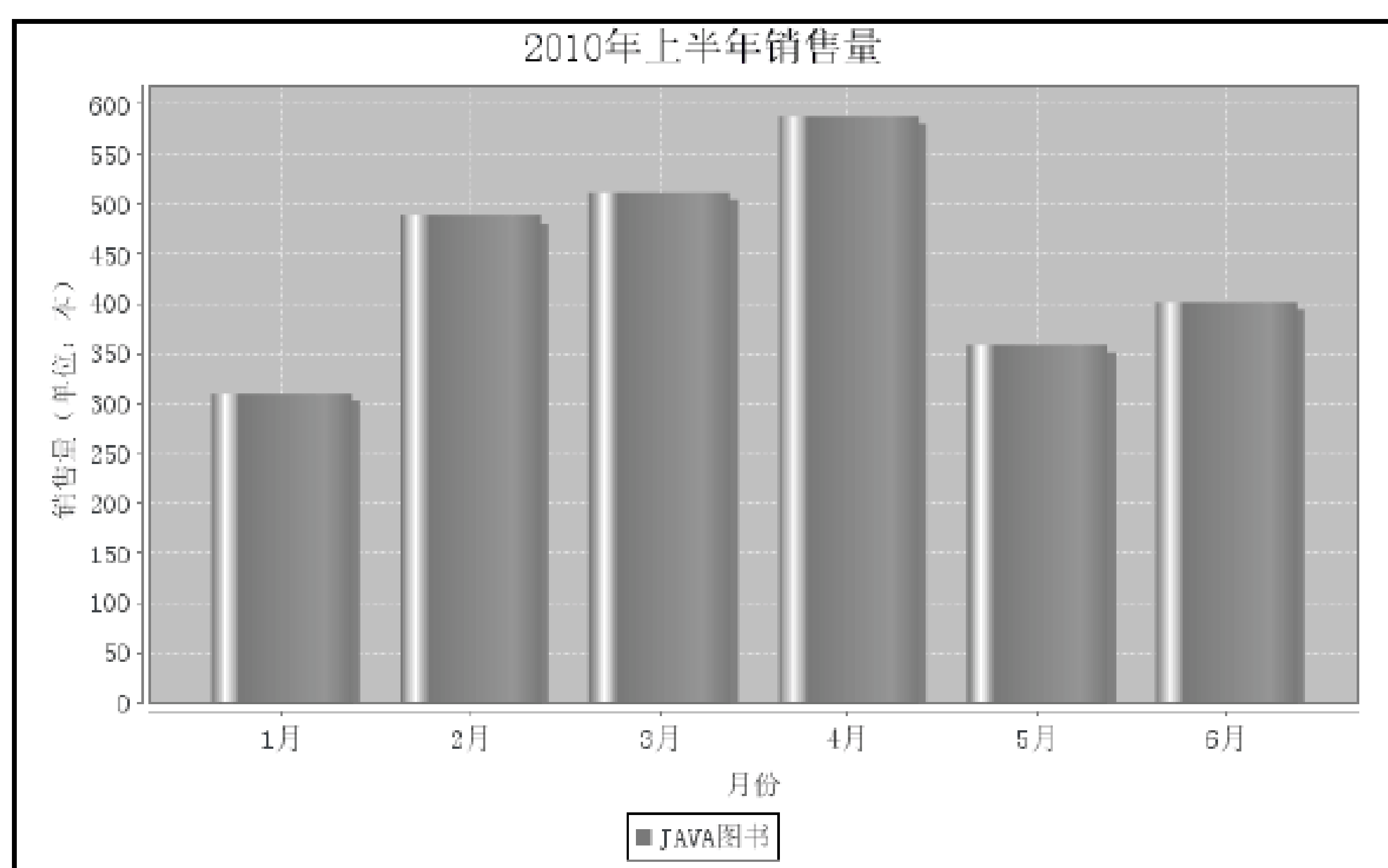


图 8.42　设置网格竖线

关键技术

使用 CategoryPlot 类的 setDomainGridlinesVisible()方法可以设置是否显示网格竖线。语法如下：

```
public void setDomainGridlinesVisible(boolean visible)
```

参数说明

visible：表示是否显示柱形图网格竖线。当 visible 为 true 时显示，为 false 时则不显示。默认值为 false。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法里获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",    //图表标题
                        "月份",                                             //X 轴标签
                        "销售量（单位：本）",                               //Y 轴标签
                        dataset,                                            //数据集
                        PlotOrientation.VERTICAL,                           //图表方向：水平、垂直
                        false,                                              //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                              //是否生成工具栏提示
                        false                                               //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 的对象，设置图像区显示网格竖线。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        CategoryPlot categoryPlot = chart.getCategoryPlot();            //图表
        categoryPlot.setDomainGridlinesVisible(true);                   //设置网格竖线显示与否
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 217：设置网格线。

默认情况下，柱形图图像区中只显示网格横向虚线，而不显示竖向虚线，除非进行手工设置。网格竖线在显示时与 X 轴的刻度一致，可以看作是 X 轴刻度的延长线。

实例 218　设置网格竖线颜色

光盘位置：光盘\MR\08\218

高级

实用指数：★★★

实例说明

柱形图网格竖线默认情况下与横线颜色相同，都为白色。本实例将网格竖线的颜色设置为蓝色，运行效果如图 8.43 所示。

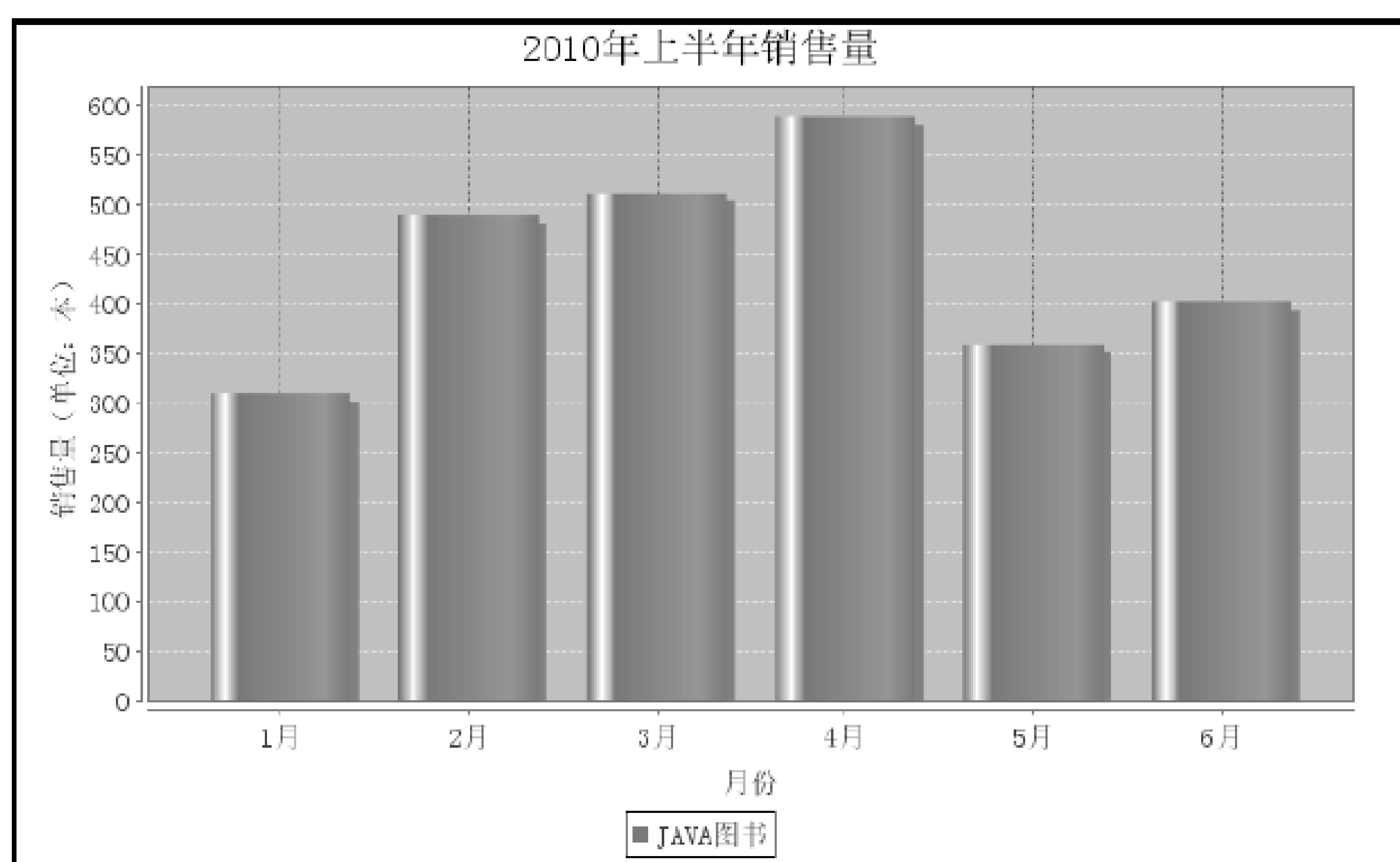


图 8.43　设置网格竖线颜色

关键技术

使用 CategoryPlot 类的 setDomainGridlinePaint()方法可以设置网格竖线的颜色。语法如下：

```
public void setDomainGridlinePaint(Paint paint)
```

参数说明

paint：表示柱形图竖线的颜色。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                "月份",                                                  //X 轴标签
                "销售量（单位：本）",                                     //Y 轴标签
                dataset,                                                 //数据集
                PlotOrientation.VERTICAL,                                //图表方向：水平、垂直
                false,                                                   //是否显示图例（对于简单的柱形图必须是 false）
                false,                                                   //是否生成工具栏提示
                false                                                    //是否生成 URL 链接
                );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 的对象，并且设置网格竖线为蓝色。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        CategoryPlot categoryPlot = chart.getCategoryPlot();             //图表
        categoryPlot.setDomainGridlinesVisible(true);
        categoryPlot.setDomainGridlinePaint(Color.blue);                 //设置网格竖线颜色
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图。具体代码参见配书光盘。

秘笈心法

心法领悟 218：修改网格竖线颜色和笔触。

图像区中的网格竖线默认情况下是不显示的，如果希望修改竖线颜色，必须先把竖线设置为可见状态。此外，还可以修改网格竖线的笔触。语法如下：

```
setDomainGridlineStroke(Stroke stroke)
```

参数说明

stroke：表示柱形图网格竖线的笔触。

实例 219　设置柱形图文本注解

光盘位置：光盘\MR\08\219

高级

实用指数：★★★★

在显示柱形图时，可以为图形添加文本注解。本实例将柱形数据通过文本注解的方式显示出来，运行效果如图 8.44 所示。

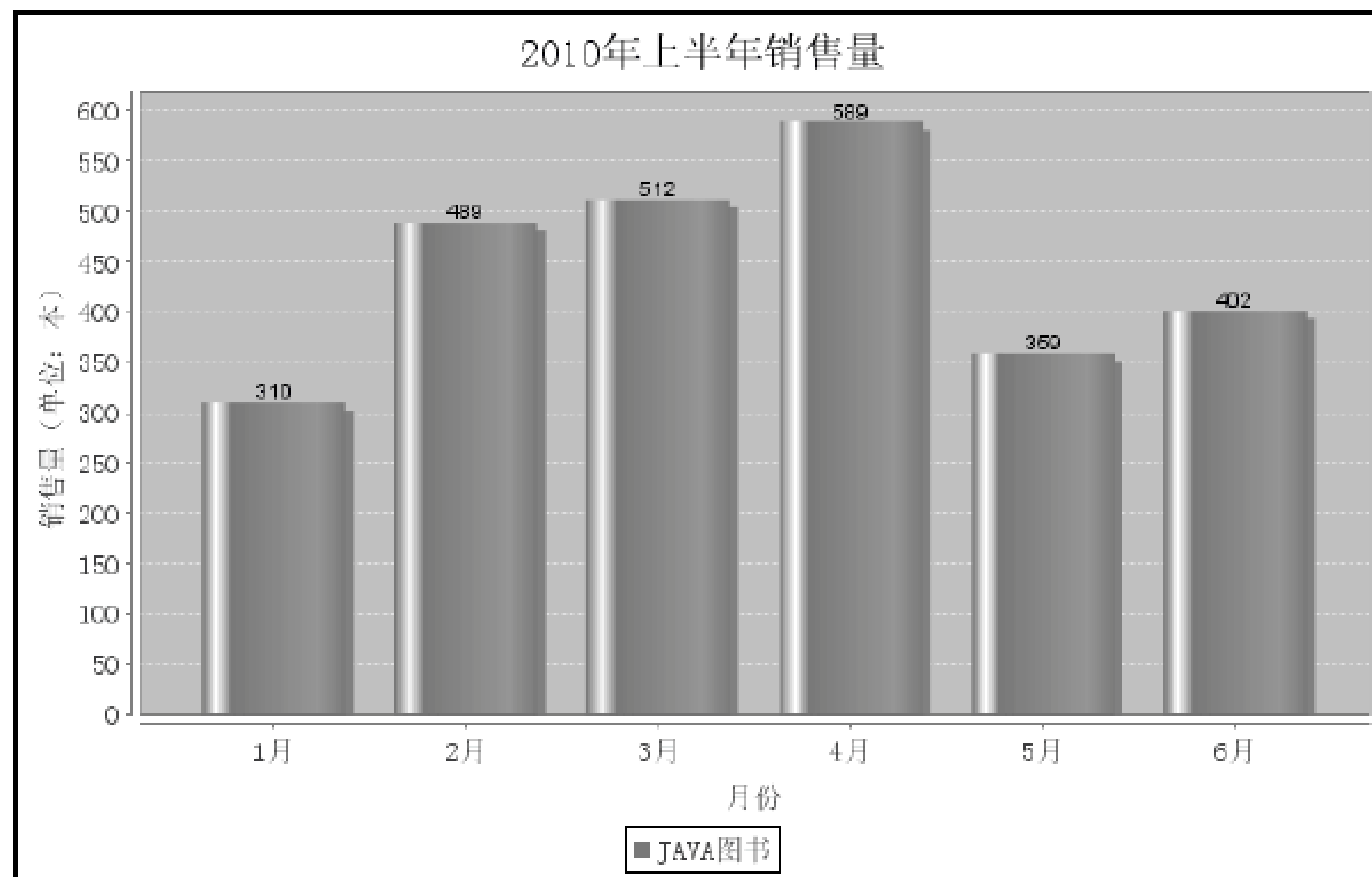


图 8.44　设置柱形图文本注解

关键技术

使用 CategoryPlot 类的 addAnnotation()方法可以为分类图形添加文本注解。语法如下：

```
public void addAnnotation(CategoryAnnotation annotation)
```

参数说明

annotation：表示柱形图的文本注解类。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "月份",                                               //X 轴标签
                        "销售量（单位：本）",                                  //Y 轴标签
                        dataset,                                              //数据集
                        PlotOrientation.VERTICAL,                             //图表方向：水平、垂直
                        false,                                                //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                                //是否生成工具栏提示
                        false                                                 //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 的对象，并且为每个分类添加文本注解，注解的内容为分类的数值。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置注解
        CategoryTextAnnotation annotation = new CategoryTextAnnotation("310","1 月",320);
        CategoryTextAnnotation annotation1 = new CategoryTextAnnotation("489","2 月",499);
        CategoryTextAnnotation annotation2 = new CategoryTextAnnotation("512","3 月",522);
        CategoryTextAnnotation annotation3 = new CategoryTextAnnotation("589","4 月",599);
        CategoryTextAnnotation annotation4 = new CategoryTextAnnotation("359","5 月",369);
        CategoryTextAnnotation annotation5 = new CategoryTextAnnotation("402","6 月",412);
        //添加注解
        categoryPlot.addAnnotation(annotation);
        categoryPlot.addAnnotation(annotation1);
        categoryPlot.addAnnotation(annotation2);
        categoryPlot.addAnnotation(annotation3);
        categoryPlot.addAnnotation(annotation4);
        categoryPlot.addAnnotation(annotation5);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 219：CategoryTextAnnotation 类。

CategoryTextAnnotation 为柱形图的文本注解类，可通过其构造方法创建 CategoryTextAnnotation 实例。语法如下：

```
CategoryTextAnnotation(String text, Comparable category, double value)
```

参数说明

❶ text：表示柱形图文本注解的内容。

❷ category：表示要添加文本注解的柱形图类别。

❸ value：表示要添加文本注解的位置。



实例 220	设置柱形图文本注解字体 光盘位置：光盘\MR\08\220	高级 实用指数：★★★

实例说明

在为柱形图添加文本注解时，如果默认的字体和颜色不符合要求，可根据实际对其进行调整。本实例将注解字体设置为“宋体”，颜色设置为白色，运行结果如图 8.45 所示。

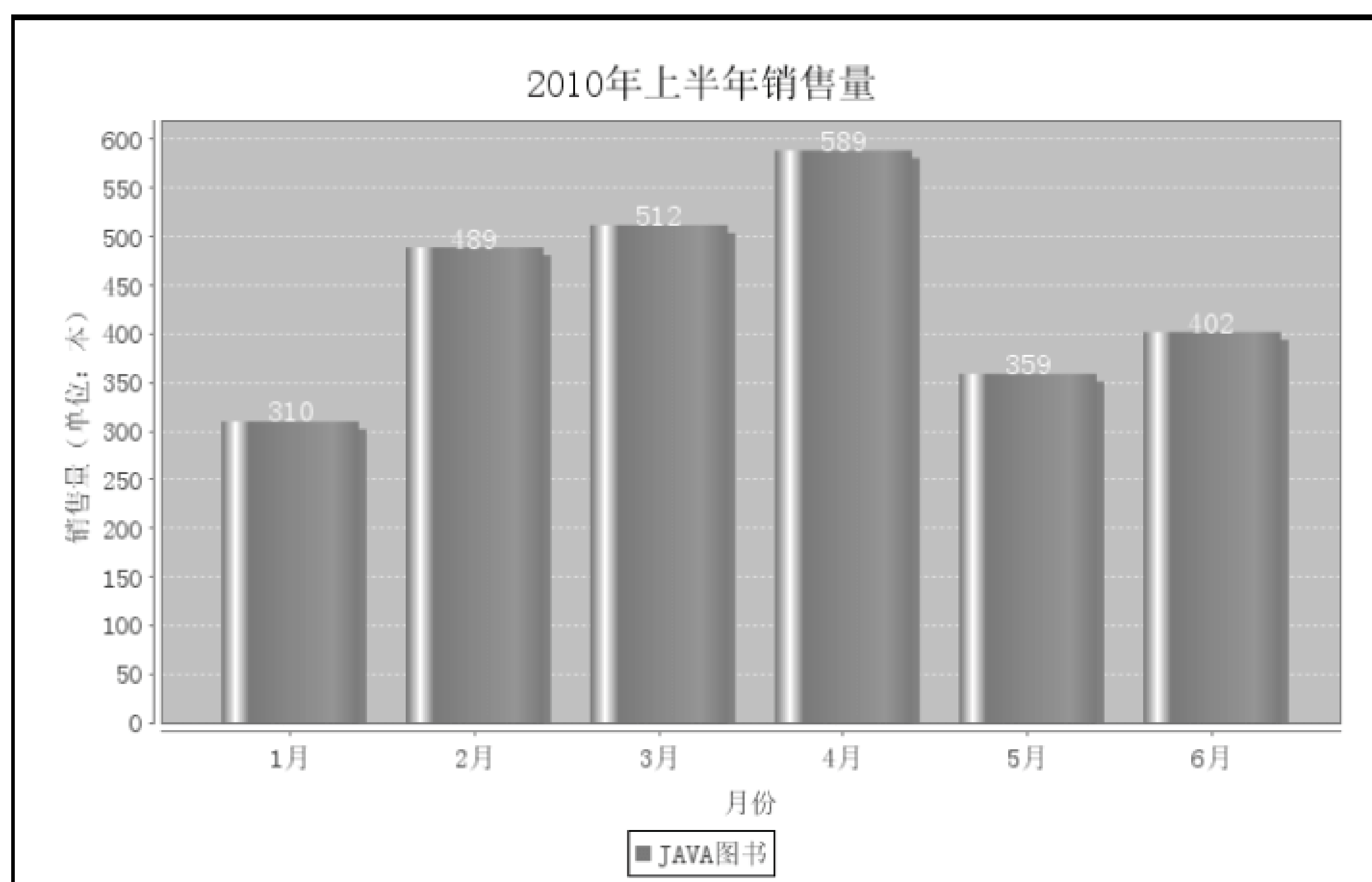


图 8.45　设置柱形图注解字体和颜色

关键技术

使用 TextAnnotation 类的 setFont()方法可以为分类图形设置文本注解字体。语法如下：

```
public void setFont(Font font)
```

参数说明

font：表示柱形图的文本注解字体。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
```

```
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "月份",                                                 //X 轴标签
                        "销售量（单位：本）",                                    //Y 轴标签
                        dataset,                                                //数据集
                        PlotOrientation.VERTICAL,                               //图表方向：水平、垂直
                        false,                                                  //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                                  //是否生成工具栏提示
                        false                                                   //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 的对象，并且为每个柱形添加文本注解，注解的内容为分类的数值，然后设置注解字体为“宋体”，其颜色为白色。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置注解
        CategoryTextAnnotation annotation = new CategoryTextAnnotation("310","1 月",320);
        CategoryTextAnnotation annotation1 = new CategoryTextAnnotation("489","2 月",499);
        CategoryTextAnnotation annotation2 = new CategoryTextAnnotation("512","3 月",522);
        CategoryTextAnnotation annotation3 = new CategoryTextAnnotation("589","4 月",599);
        CategoryTextAnnotation annotation4 = new CategoryTextAnnotation("359","5 月",369);
        CategoryTextAnnotation annotation5 = new CategoryTextAnnotation("402","6 月",412);
        //设置注解字体
        annotation.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation1.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation2.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation3.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation4.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation5.setFont(new Font("宋体", Font.PLAIN, 15));
        //设置注解颜色
        annotation.setPaint(Color.WHITE);
        annotation1.setPaint(Color.WHITE);
        annotation2.setPaint(Color.WHITE);
        annotation3.setPaint(Color.WHITE);
        annotation4.setPaint(Color.WHITE);
        annotation5.setPaint(Color.WHITE);
        //添加注解
        categoryPlot.addAnnotation(annotation);
        categoryPlot.addAnnotation(annotation1);
        categoryPlot.addAnnotation(annotation2);
        categoryPlot.addAnnotation(annotation3);
        categoryPlot.addAnnotation(annotation4);
        categoryPlot.addAnnotation(annotation5);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

心法领悟 220：为柱形图文本注解设置字体大小。

为柱形图文本注解设置字体时，修改字体大小可能会影响字体与图形之间的距离，所以有时要根据字体调整文本注解在图形中的位置。



实例 221　**设置柱形图文本注解锚点**　高级

光盘位置：光盘\MR\08\221　实用指数：★★★

实例说明

柱形图的文本注解在实例化时可以设置垂直高度，实例化以后还可以设置文本注解锚点。本例将演示如何设置柱形图文本注解锚点，运行效果如图 8.46 所示。

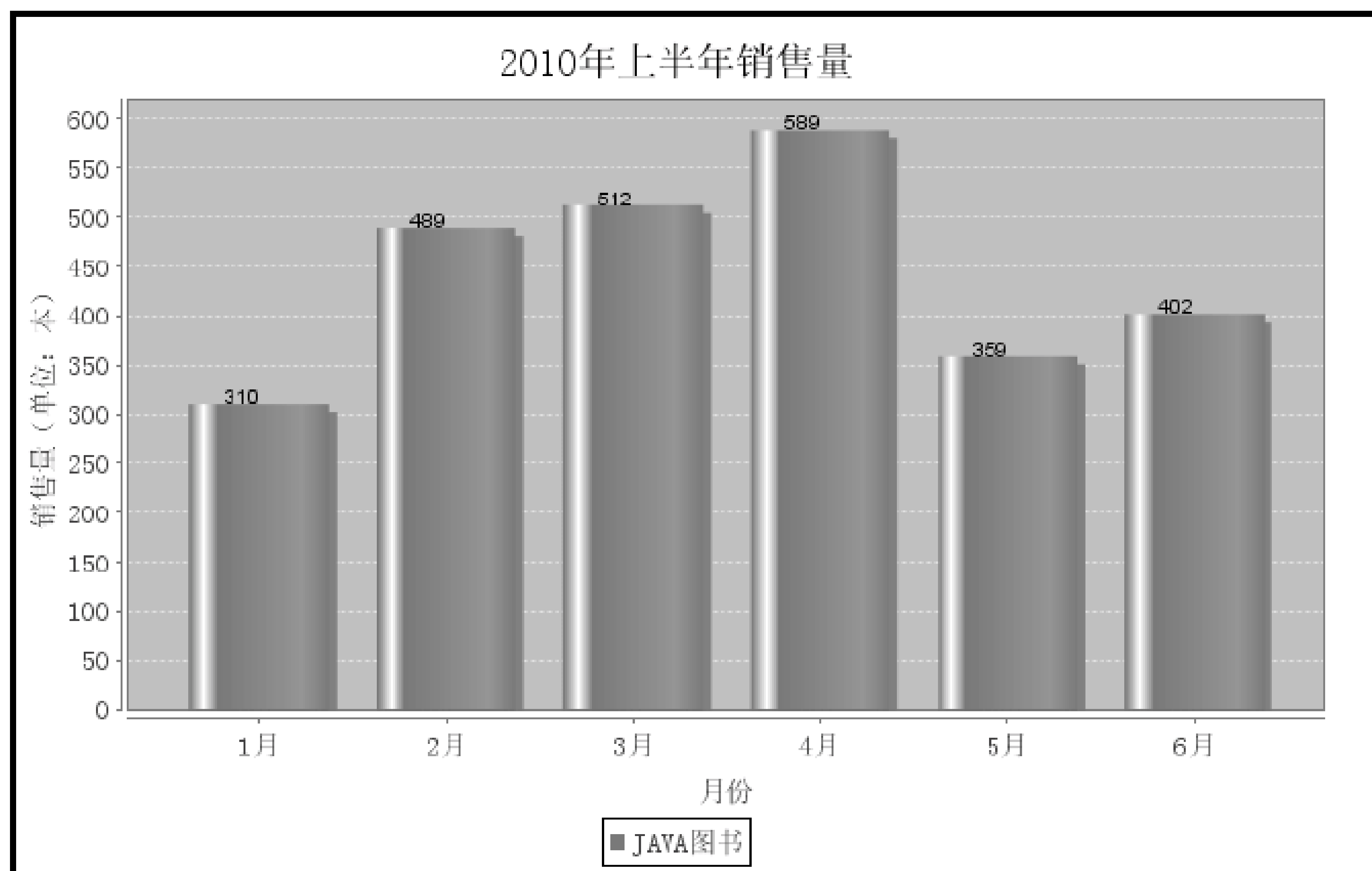


图 8.46　设置柱形图文本注解锚点

关键技术

使用 TextAnnotation 类的 setTextAnchor()方法可以为分类图形设置文本注解锚点，调整文本的对齐方式。语法如下：

```
public void setTextAnchor(TextAnchor anchor)
```

参数说明

anchor：表示柱形图注解的文本锚点。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码

如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",  //图表标题
                    "月份",                                     //X 轴标签
                    "销售量（单位：本）",                          //Y 轴标签
                    dataset,                                    //数据集
                    PlotOrientation.VERTICAL,                   //图表方向：水平、垂直
                    false,                                      //是否显示图例（对于简单的柱形图必须是 false）
                    false,                                      //是否生成工具栏提示
                    false                                       //是否生成 URL 链接
                    );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 的对象，并且为每个柱形设置文本注解锚点。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置注解
        CategoryTextAnnotation annotation = new CategoryTextAnnotation("310","1 月",310);
        CategoryTextAnnotation annotation1 = new CategoryTextAnnotation("489","2 月",489);
        CategoryTextAnnotation annotation2 = new CategoryTextAnnotation("512","3 月",512);
        CategoryTextAnnotation annotation3 = new CategoryTextAnnotation("589","4 月",589);
        CategoryTextAnnotation annotation4 = new CategoryTextAnnotation("359","5 月",359);
        CategoryTextAnnotation annotation5 = new CategoryTextAnnotation("402","6 月",402);
        //设置文本注解锚点
        annotation.setTextAnchor(TextAnchor.BASELINE_RIGHT);
        annotation1.setTextAnchor(TextAnchor.BASELINE_RIGHT);
        annotation2.setTextAnchor(TextAnchor.BASELINE_RIGHT);
        annotation3.setTextAnchor(TextAnchor.BASELINE_RIGHT);
        annotation4.setTextAnchor(TextAnchor.BASELINE_RIGHT);
        annotation5.setTextAnchor(TextAnchor.BASELINE_RIGHT);
        //添加注解
        categoryPlot.addAnnotation(annotation);
        categoryPlot.addAnnotation(annotation1);
        categoryPlot.addAnnotation(annotation2);
        categoryPlot.addAnnotation(annotation3);
        categoryPlot.addAnnotation(annotation4);
        categoryPlot.addAnnotation(annotation5);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 221：TextAnchor 类。

TextAnchor 类实现了很多文本注解锚点位置的常量，如 TextAnchor.CENTER、TextAnchor.BASELINE_CENTER、TextAnchor.BASELINE_LEFT、TextAnchor.BASELINE_RIGHT、TextAnchor.BOTTOM_CENTER 等。

实例 222 设置柱形图文本注解的类别锚点

光盘位置：光盘\MR\08\222

高级

实用指数：★★★

实例说明

柱形图的文本注解可以在实例化时设置垂直位置，实例化以后还可以设置文本注解的类别锚点。本实例将演示如何设置柱形图文本注解的类别锚点，运行结果如图 8.47 所示。

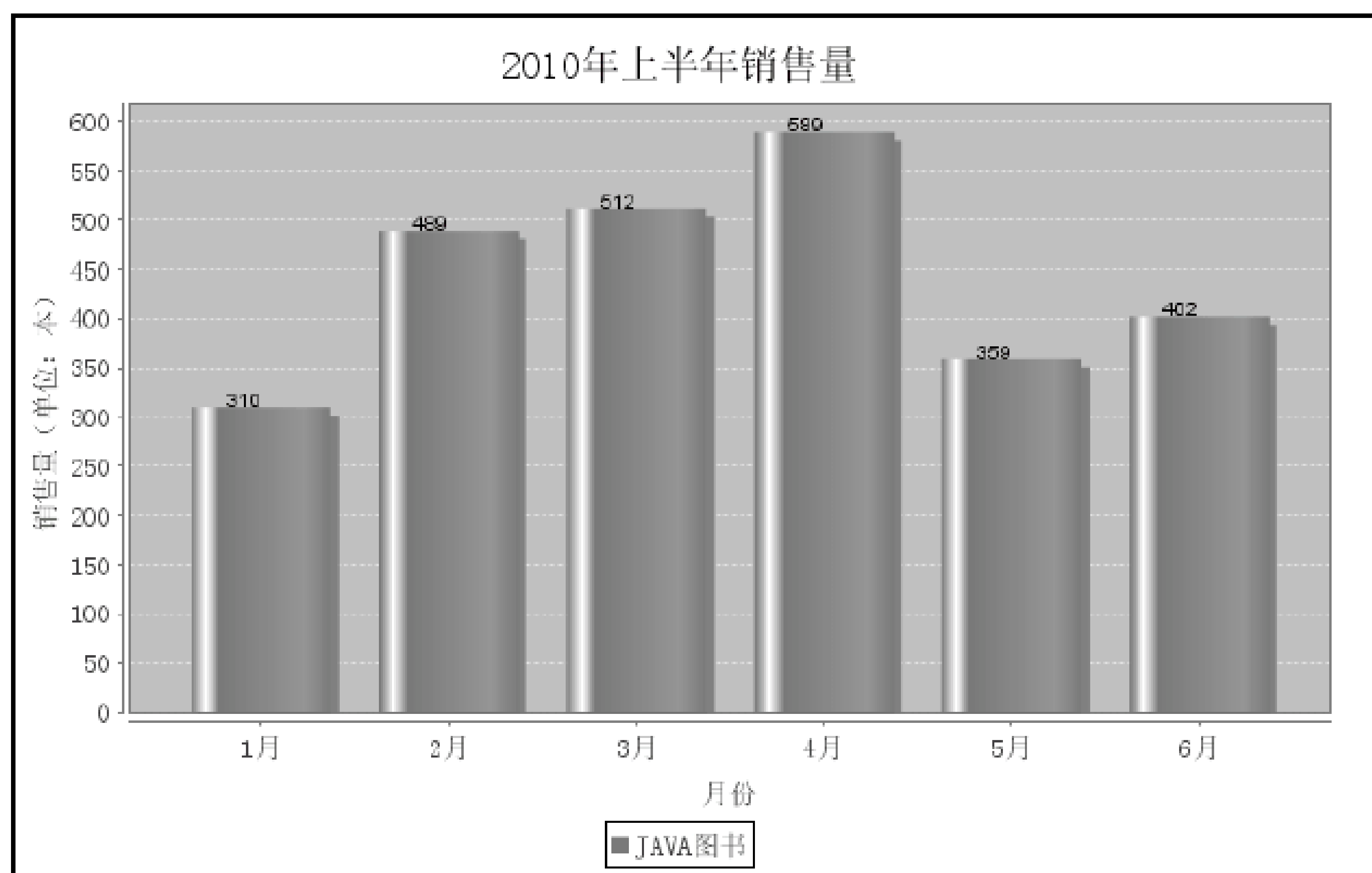


图 8.47　设置柱形图文本注解的类别锚点

关键技术

TextAnnotation 类一般用于文本的注解中；CategoryTextAnnotation 类一般用于类别注解。利用 CategoryTextAnnotation 类的 setCategoryAnchor()方法可以设置类别锚点。语法如下：

```
public void setCategoryAnchor(CategoryAnchor anchor)
```

参数说明

anchor：表示柱形图文本注解的类别锚点。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                    "月份",                                           //X 轴标签
                    "销售量（单位：本）",                                //Y 轴标签
                    dataset,                                          //数据集
                    PlotOrientation.VERTICAL,                         //图表方向：水平、垂直
                    false,                                            //是否显示图例（对于简单的柱形图必须是 false）
                    false,                                            //是否生成工具栏提示
                    false                                             //是否生成 URL 链接
                    );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 的对象，并且为每个柱形文本注解设置类别锚点。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置注解
        CategoryTextAnnotation annotation = new CategoryTextAnnotation("310","1 月",310);
        CategoryTextAnnotation annotation1 = new CategoryTextAnnotation("489","2 月",489);
        CategoryTextAnnotation annotation2 = new CategoryTextAnnotation("512","3 月",512);
        CategoryTextAnnotation annotation3 = new CategoryTextAnnotation("589","4 月",589);
        CategoryTextAnnotation annotation4 = new CategoryTextAnnotation("359","5 月",359);
        CategoryTextAnnotation annotation5 = new CategoryTextAnnotation("402","6 月",402);
        //设置注解类别锚点
        annotation.setCategoryAnchor(CategoryAnchor.END);
        annotation1.setCategoryAnchor(CategoryAnchor.END);
        annotation2.setCategoryAnchor(CategoryAnchor.END);
        annotation3.setCategoryAnchor(CategoryAnchor.END);
        annotation4.setCategoryAnchor(CategoryAnchor.END);
        annotation5.setCategoryAnchor(CategoryAnchor.END);
        //添加注解
        categoryPlot.addAnnotation(annotation);
        categoryPlot.addAnnotation(annotation1);
        categoryPlot.addAnnotation(annotation2);
        categoryPlot.addAnnotation(annotation3);
        categoryPlot.addAnnotation(annotation4);
        categoryPlot.addAnnotation(annotation5);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 222：CategoryAnchor 类的常量。

JFreeChart 在 CategoryAnchor 类中定义了 3 个常量，分别为 CategoryAnchor.START、CategoryAnchor.MIDDLE 和 CategoryAnchor.END，用于表示柱形图文本注解的类别锚点。

实例 223 设置柱形图文本注解的旋转锚点

光盘位置：光盘\MR\08\223

高级

实用指数：★★

柱形图的文本注解可以在实例化时设置垂直位置，实例化以后还可以设置文本注解的旋转锚点。本实例将演示如何设置柱形图文本注解的旋转锚点，运行结果如图 8.48 所示。

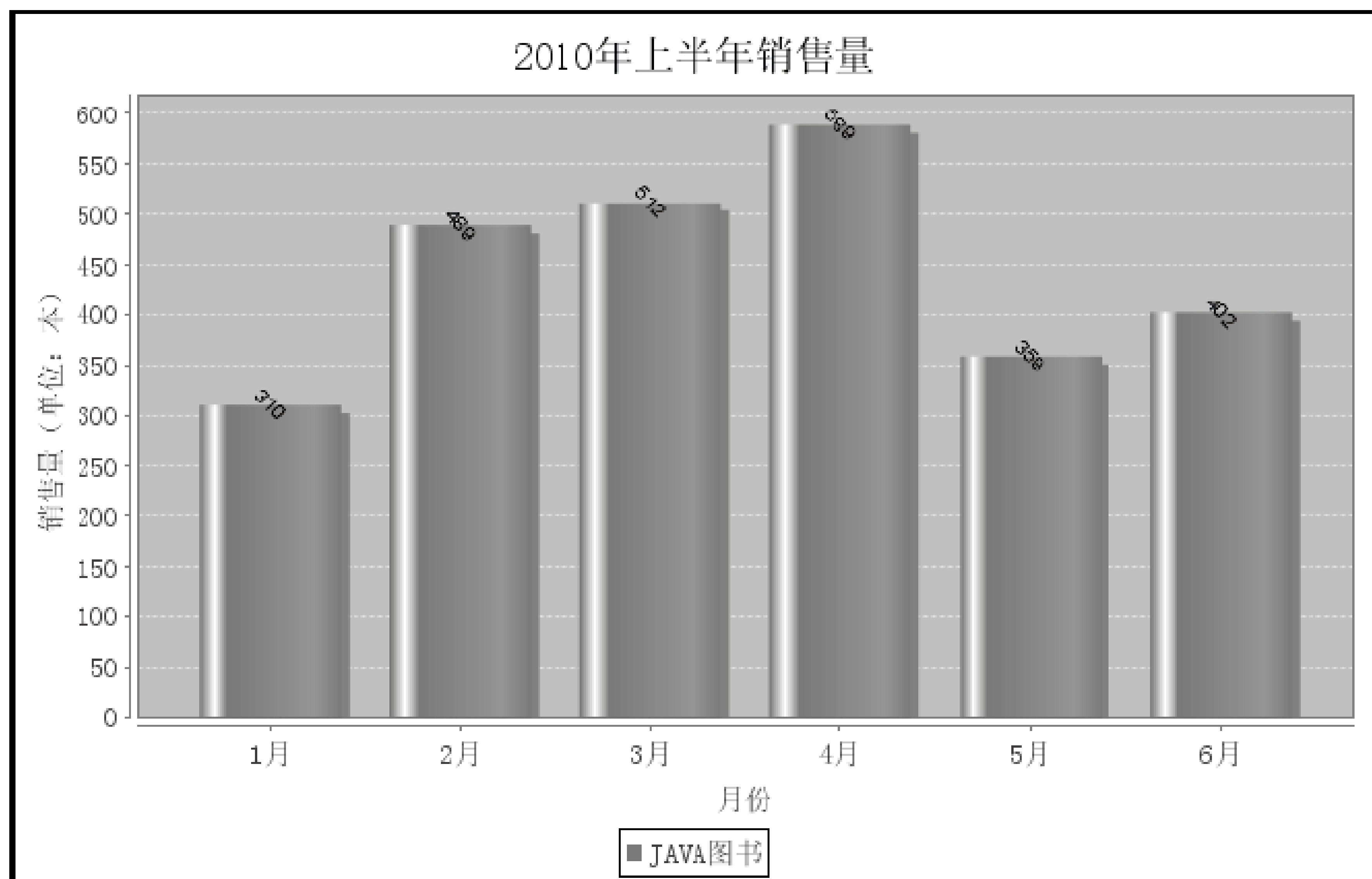


图 8.48　设置柱形图文本注解的旋转锚点

关键技术

使用 TextAnnotation 的 setRotationAngle()方法可以设置文本注解的旋转锚点。语法如下：

```
public void setRotationAngle(double angle)
```

参数说明

angle：表示柱形图文本注解的旋转锚点。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",    //图表标题
                        "月份",                                                  //X 轴标签
                        "销售量（单位：本）",                                    //Y 轴标签
                        dataset,                                                 //数据集
                        PlotOrientation.VERTICAL,                                //图表方向：水平、垂直
                        false,                                                   //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                                   //是否生成工具栏提示
                        false                                                    //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 的对象，并且为每个柱形文本注解设置旋转锚点，参数值为 Math.PI*0.2（其中的 Math.PI 为圆周率）。代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置注解
        CategoryTextAnnotation annotation = new CategoryTextAnnotation("310","1 月",310);
        CategoryTextAnnotation annotation1 = new CategoryTextAnnotation("489","2 月",489);
        CategoryTextAnnotation annotation2 = new CategoryTextAnnotation("512","3 月",512);
        CategoryTextAnnotation annotation3 = new CategoryTextAnnotation("589","4 月",589);
        CategoryTextAnnotation annotation4 = new CategoryTextAnnotation("359","5 月",359);
        CategoryTextAnnotation annotation5 = new CategoryTextAnnotation("402","6 月",402);
        //设置注解旋转锚点
        annotation.setRotationAngle(Math.PI*0.2);
        annotation1.setRotationAngle(Math.PI*0.2);
        annotation2.setRotationAngle(Math.PI*0.2);
        annotation3.setRotationAngle(Math.PI*0.2);
        annotation4.setRotationAngle(Math.PI*0.2);
        annotation5.setRotationAngle(Math.PI*0.2);
        //添加注解
        categoryPlot.addAnnotation(annotation);
        categoryPlot.addAnnotation(annotation1);
```

```
        categoryPlot.addAnnotation(annotation2);
        categoryPlot.addAnnotation(annotation3);
        categoryPlot.addAnnotation(annotation4);
        categoryPlot.addAnnotation(annotation5);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 223：设置旋转锚点。

JFreeChart 柱形图文本注解使用数值设置的旋转锚点，通过数值的大小来表示旋转的角度。其中，锚点的数值指的是弧度数，PI 弧度角等于 180°，所以本实例使用 PI*0.2 就是设置文本注解的旋转角度为 36°。

实例 224 设置柱形图线条注解

光盘位置：光盘\MR\08\224

高级

实用指数：★★★★

实例说明

线条注解是一条直线，通过它可以把任意两个柱形连接在一起，从而在柱形图上体现出折线图的效果。本实例将演示如何设置柱形图线条注解，运行结果如图 8.49 所示。

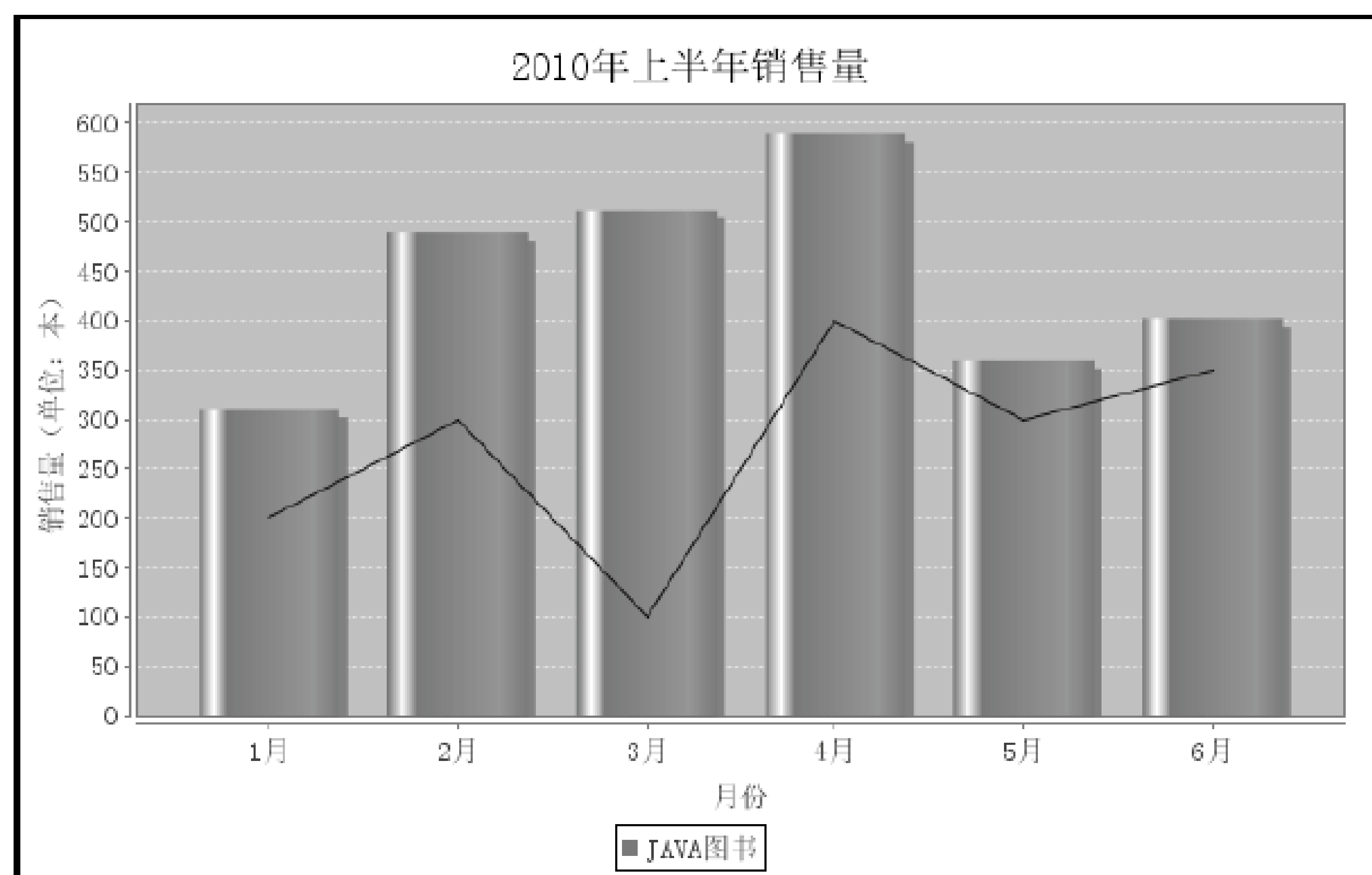


图 8.49　设置柱形图线条注解

关键技术

使用 TextAnnotation 类的 addAnnotation()方法可以设置线条注解。语法如下：

```
public void addAnnotation(CategoryAnnotation annotation)
```

参数说明

annotation：表示要设置的柱形图的线条注解。

（1）创建 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
```

```
		//创建数据集合实例
		CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
		return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
private JFreeChart getJFreeChart() {
		CategoryDataset dataset = getCategoryDataset();
		JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",	//图表标题
				"月份",								//X 轴标签
				"销售量（单位：本）",						//Y 轴标签
				dataset,								//数据集
				PlotOrientation.VERTICAL,					//图表方向：水平、垂直
				false,								//是否显示图例（对于简单的柱形图必须是 false）
				false,								//是否生成工具栏提示
				false								//是否生成 URL 链接
				);
		return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 的对象，并且为柱形图设置线条注解，绘制一个折线图。代码如下：

```
public static void updatePlot(JFreeChart chart) {
		//图表
		CategoryPlot categoryPlot = chart.getCategoryPlot();
		//设置线条注解
		CategoryLineAnnotation annotation = new CategoryLineAnnotation("1 月",200,"2 月",300,Color.blue,new BasicStroke());
		CategoryLineAnnotation annotation1 = new CategoryLineAnnotation("2 月",300,"3 月",100,Color.blue,new BasicStroke());
		CategoryLineAnnotation annotation2 = new CategoryLineAnnotation("3 月",100,"4 月",400,Color.blue,new BasicStroke());
		CategoryLineAnnotation annotation3 = new CategoryLineAnnotation("4 月",400,"5 月",300,Color.blue,new BasicStroke());
		CategoryLineAnnotation annotation4 = new CategoryLineAnnotation("5 月",300,"6 月",350,Color.blue,new BasicStroke());
		categoryPlot.addAnnotation(annotation);
		categoryPlot.addAnnotation(annotation1);
		categoryPlot.addAnnotation(annotation2);
		categoryPlot.addAnnotation(annotation3);
		categoryPlot.addAnnotation(annotation4);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 224：CategoryLineAnnotation 类。

绘制线条注解要使用 CategoryLineAnnotation 类创建一个实例，通过其构造方法创建实例非常简单，语法如下：

```
CategoryLineAnnotation(Comparable category1, double value1,Comparable category2, double value2, Paint paint, Stroke stroke)
```

参数说明

❶ category1：表示前一个柱形图的分类名称。

❷ value1：表示前一个柱形图的线条注解的垂直高度。

❸ category2：表示后一个柱形图的分类名称。

❹ value2：表示后一个柱形图的线条注解的垂直高度。

❺ paint：表示线条注解的颜色。

❻ stroke：表示线条注解的笔触。



实例 225	绘制柱形效果 光盘位置：光盘\MR\08\225	高级 实用指数：★★★

实例说明

JFreeChart 在生成柱形图时，可以根据需要显示柱形的效果。本实例通过 JFreeChart 提供的 BarRenderer 类，

使用普通的平面效果显示柱形图，如图 8.50 所示。

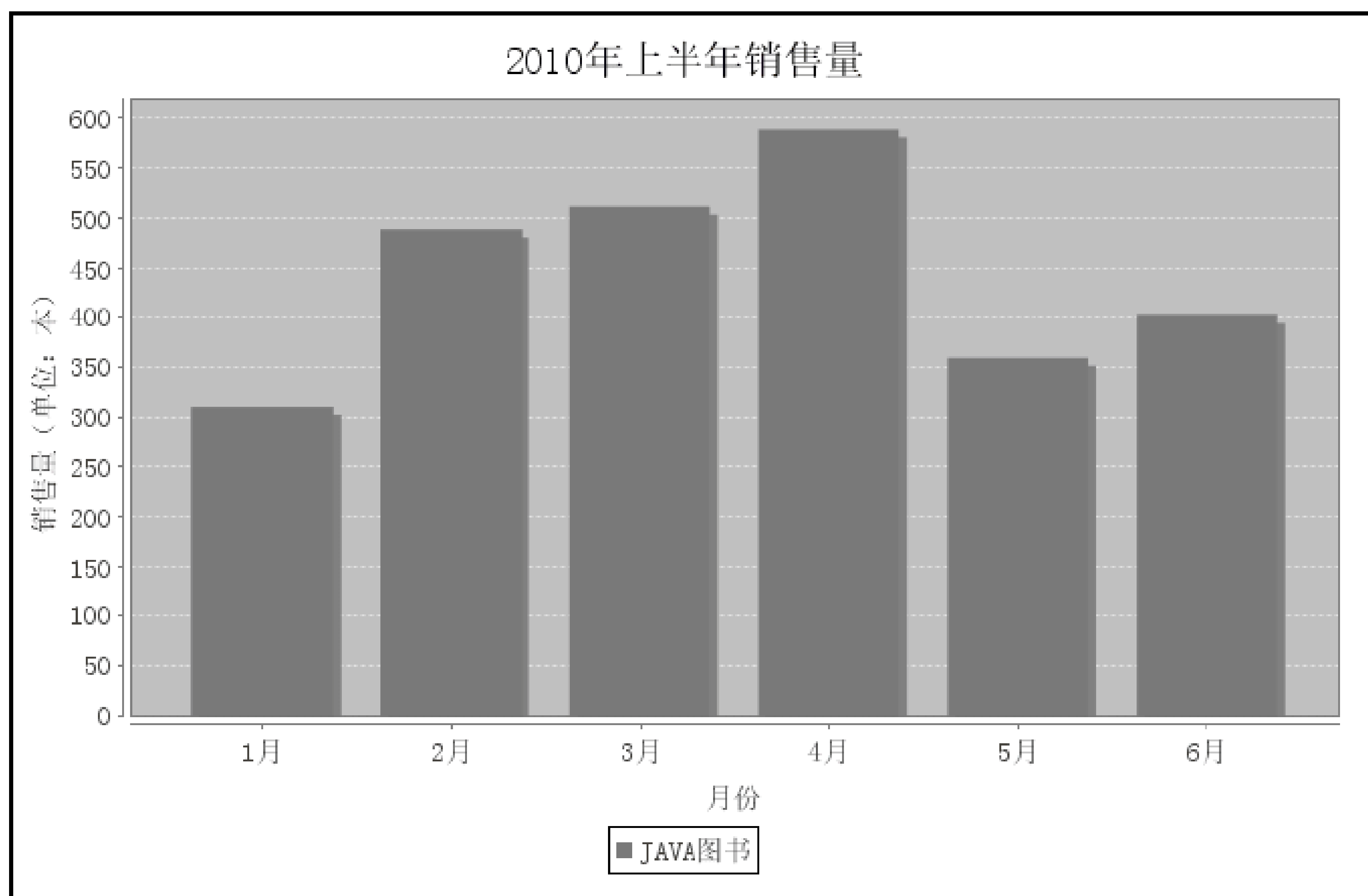


图 8.50　标准的柱形效果

关键技术

使用 BarRenderer 类的 setBarPainter()方法可以设置柱形效果。语法如下：

```
public void setBarPainter(BarPainter painter)
```

参数说明

painter：表示要绘制的柱形效果。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量 ",        //图表标题
                        "月份",                                        //X 轴标签
                        "销售量（单位：本）",                          //Y 轴标签
                        dataset,                                       //数据集
                        PlotOrientation.VERTICAL,                      //图表方向：水平、垂直
                        false,                                         //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                         //是否生成工具栏提示
                        false                                          //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 的对象，并且绘制一个普通的柱形效果图。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置 Y 轴显示位置
        BarRenderer renderer = (BarRenderer) categoryPlot.getRenderer();
        //普通效果
        StandardBarPainter barPainter = new StandardBarPainter();
        //梯形效果
        GradientBarPainter barPainter = new GradientBarPainter();
        renderer.setBarPainter(barPainter);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 225：StandardBarPainter 类与 GradientBarPainter 类。

StandardBarPainter 类与 GradientBarPainter 类都继承了 BarPainter 接口，实现了 BarPainter 中抽象的方法；其不同之处在于 StandardBarPainter 类绘制了柱形图的普通效果，而 GradientBarPainter 类绘制了柱形图呈梯形的立体效果。

实例 226　柱形图阴影　高级

光盘位置：光盘\MR\08\226　实用指数：★★

实例说明

柱形图在显示时，如果设置阴影效果可以使图表更生动。用户可以根据需要决定是否使用阴影效果。本实例将演示如何取消阴影效果，运行结果如图 8.51 所示。

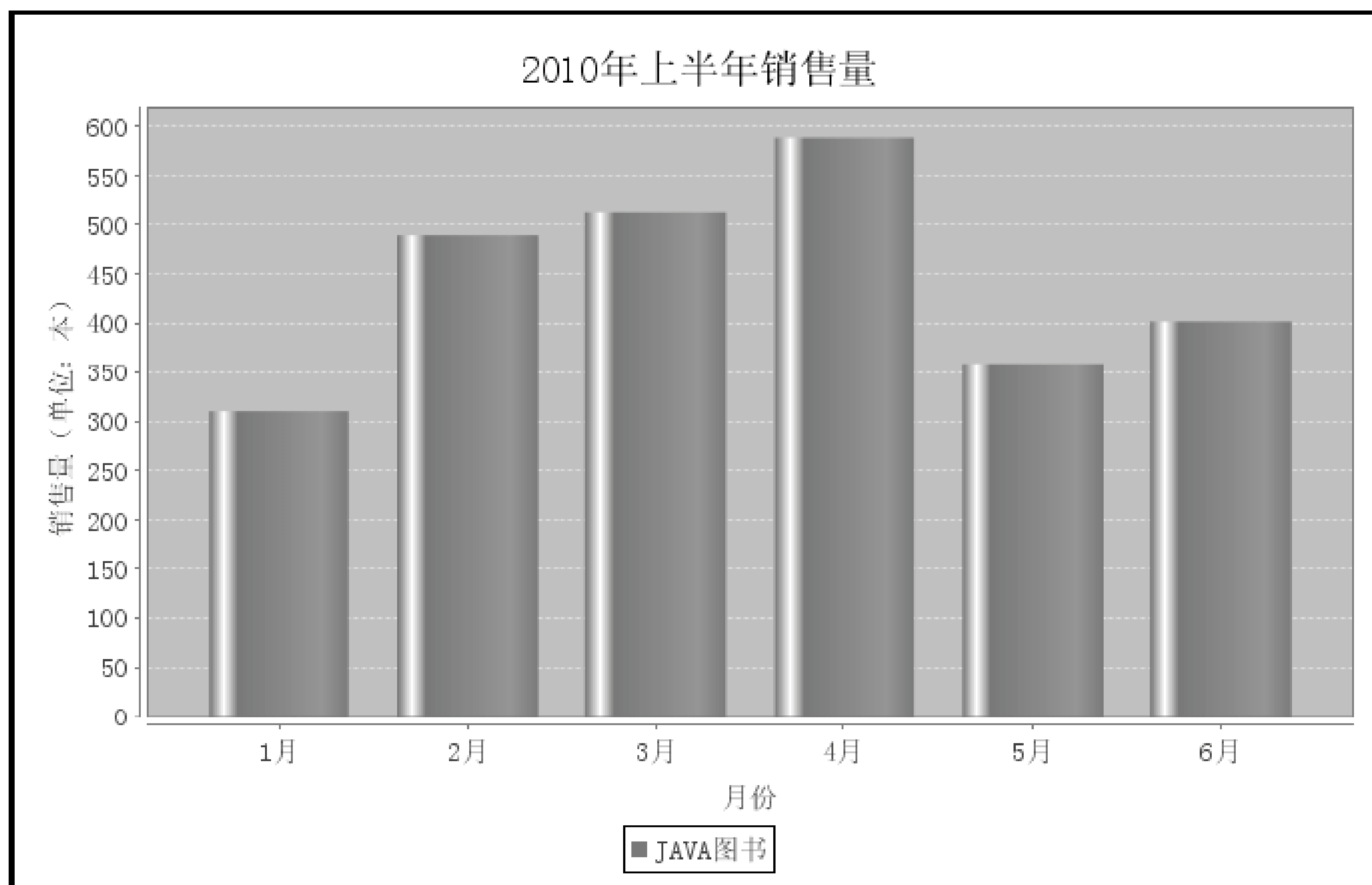


图 8.51　取消阴影效果

使用 BarRenderer 类的 setShadowVisible()方法可以设置柱形图阴影效果。语法如下：

```
public void setShadowVisible(boolean visible)
```

参数说明

visible：表示是否显示阴影效果。

设计过程

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "月份",                                          //X 轴标签
                        "销售量（单位：本）",                            //Y 轴标签
                        dataset,                                         //数据集
                        PlotOrientation.VERTICAL,                        //图表方向：水平、垂直
                        false,                                           //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                           //是否生成工具栏提示
                        false                                            //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 的对象，通过 CategoryPlot 的对象获取 BarRenderer 对象，然后设置隐藏阴影效果。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        BarRenderer renderer = (BarRenderer) categoryPlot.getRenderer();
        //阴影效果
        renderer.setShadowVisible(false);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 226：修改阴影的颜色。

默认情况下柱形图的阴影效果都是显示的，显示颜色为灰色。可以使用 BarRenderer 类的 setShadowPaint()方法修改阴影的显示颜色，语法如下：

```
setShadowPaint(Paint paint)
```

参数说明

paint：表示柱形图的阴影颜色。

实例 227	柱形图阴影偏移 光盘位置：光盘\MR\08\227	高级 实用指数：★★★

实例说明

柱形图的阴影效果可以自由控制，如本实例增加了柱形图的阴影偏移，使阴影的效果更明显，如图 8.52 所示。

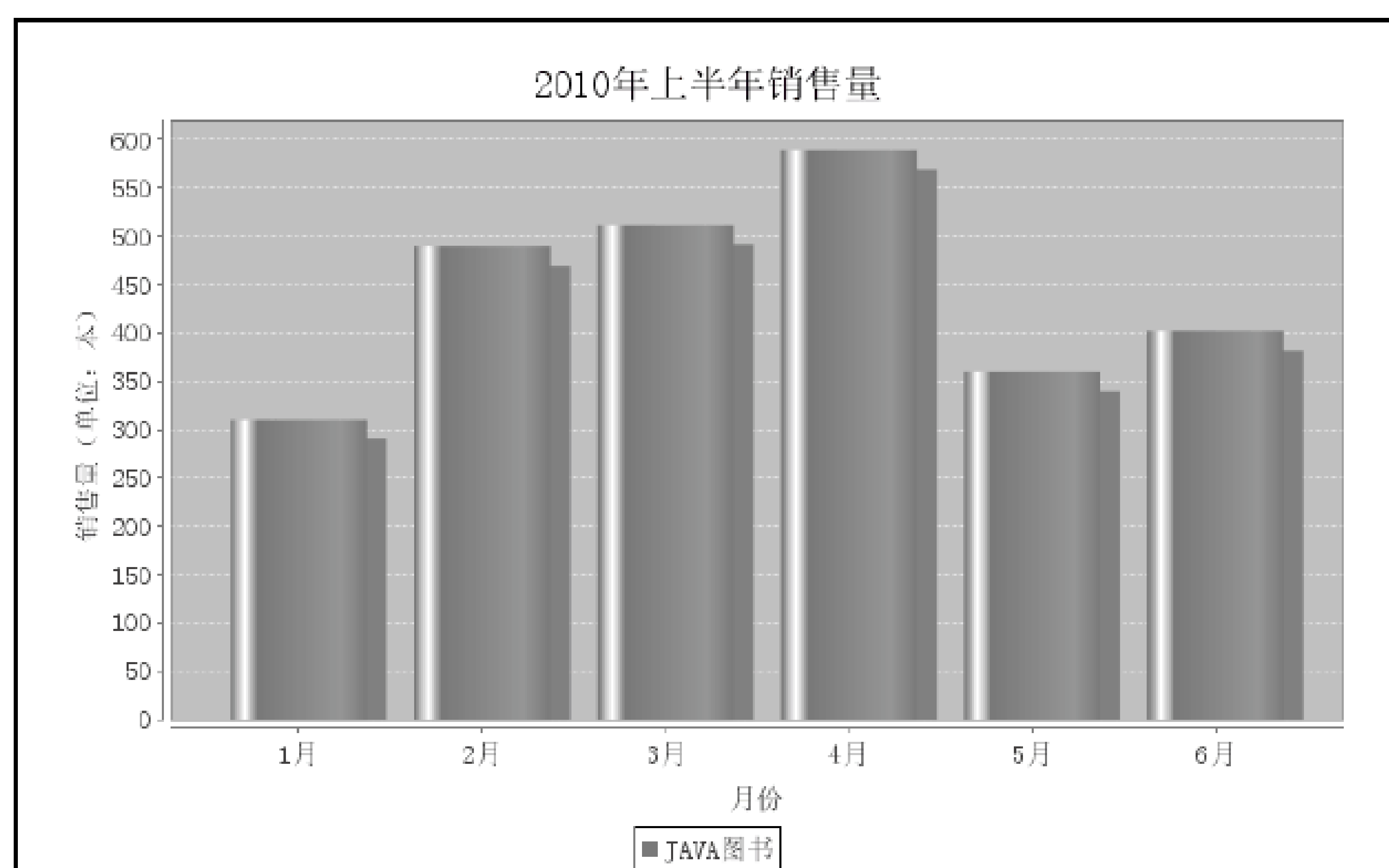


图 8.52　阴影偏移

关键技术

使用 BarRenderer 类的 setShadowXOffset()方法和 setShadowYOffset()方法可以共同设置柱形偏移效果。语法分别如下：

```
public void setShadowXOffset(double offset)
```

参数说明

offset：表示柱形 X 轴方向阴影的偏移量。

```
public void setShadowYOffset(double offset)
```

参数说明

offset：表示柱形 Y 轴方向阴影的偏移量。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                        "月份",                                               //X 轴标签
                        "销售量（单位：本）",                                   //Y 轴标签
                        dataset,                                              //数据集
                        PlotOrientation.VERTICAL,                             //图表方向：水平、垂直
                        false,                                                //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                                //是否生成工具栏提示
                        false                                                 //是否生成 URL 链接
                        );
        return chart;
```

```
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 的对象，通过 CategoryPlot 的对象获取 BarRenderer 对象，然后设置阴影的偏移效果。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        BarRenderer renderer = (BarRenderer) categoryPlot.getRenderer();
        //阴影效果
        renderer.setShadowVisible(true);
        //X 轴偏移量
        renderer.setShadowXOffset(10);
        //Y 轴偏移量
        renderer.setShadowYOffset(10);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 227：阴影的偏移量。

阴影的偏移是以柱形图中柱形的位置为基础，X 轴、Y 轴的偏移值默认情况下都为 4.0，偏移的值越大，阴影的范围越大，偏移的值越小，阴影的范围则越小。



实例 228 设置柱形的颜色 高级

光盘位置：光盘\MR\08\228 实用指数：★★★★

实例说明

在柱形图中，柱形颜色是可以设置的。本实例将演示如何设置柱形图的柱形颜色，运行结果如图 8.53 所示。

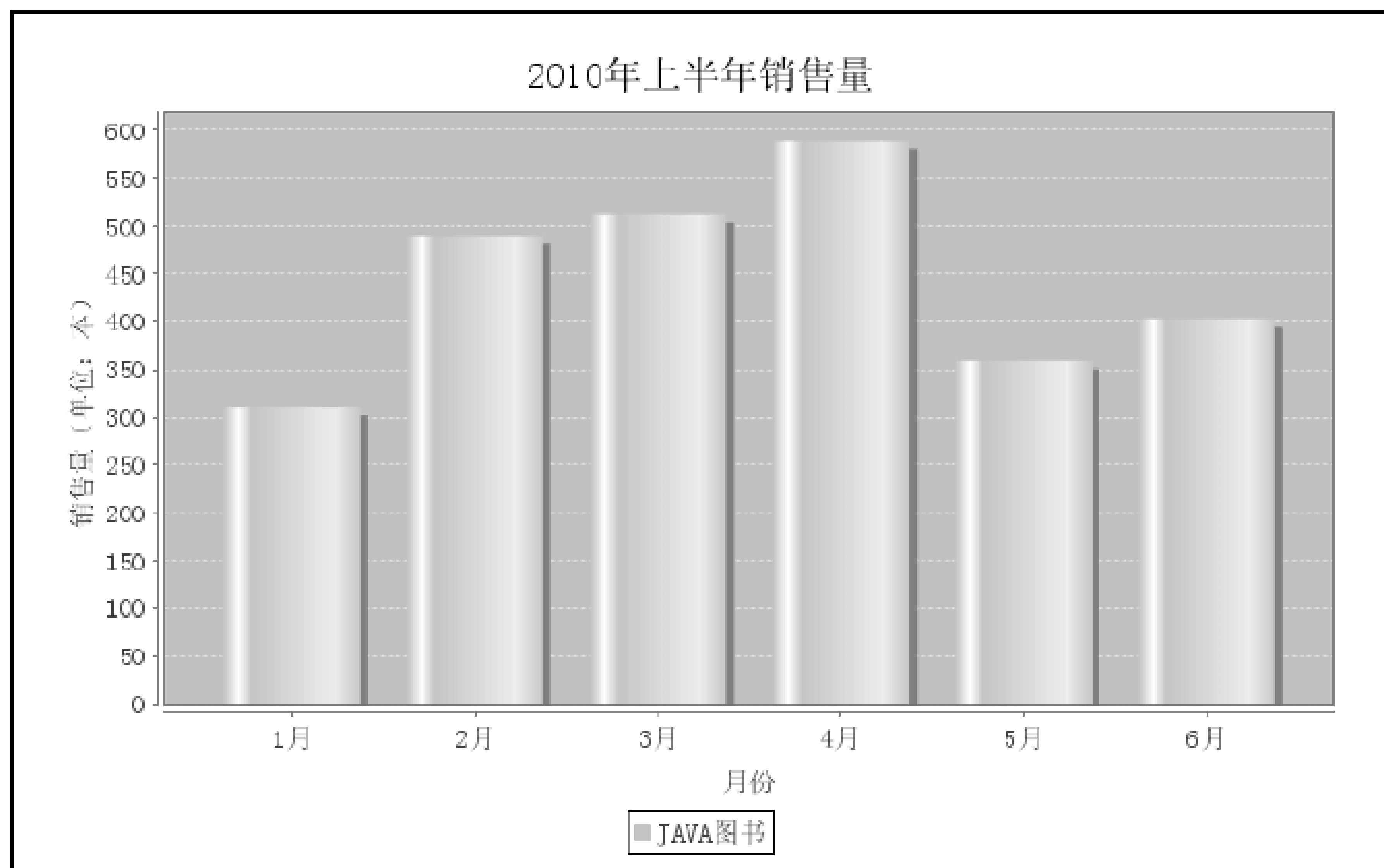


图 8.53 设置柱形图的柱形颜色

关键技术

使用 AbstractRenderer 类的 setSeriesPaint()方法可以设置柱形的颜色。语法如下：

```
public void setSeriesPaint(int series, Paint paint)
```

参数说明

❶ series：表示要设置的柱形颜色的系列。

❷ paint：表示要设置的柱形的颜色。

设计过程

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建一个适合普通柱形图的数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",   //图表标题
                    "月份",                                       //X 轴标签
                    "销售量（单位：本）",                          //Y 轴标签
                    dataset,                                      //数据集
                    PlotOrientation.VERTICAL,                     //图表方向：水平、垂直
                    false,                                        //是否显示图例（对于简单的柱形图必须是 false）
                    false,                                        //是否生成工具栏提示
                    false                                         //是否生成 URL 链接
                    );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 的对象，通过 CategoryPlot 的对象获取 BarRenderer 对象，然后为柱形设置颜色。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        BarRenderer renderer = (BarRenderer) categoryPlot.getRenderer();
        //柱形颜色
        renderer.setSeriesPaint(0, Color.orange);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 228：为柱形设置颜色时的注意事项。

为柱形设置颜色时尽量不要与背景色相近，否则将无法分辨柱形的高度，影响查看图表的效果。

实例 229	绘制 3D 柱形图 光盘位置：光盘\MR\08\229	高级 实用指数：★★★★

实例说明

普通的柱形图显示的效果始终不够立体，本实例将演示如何绘制 3D 效果的柱形图，运行结果如图 8.54 所示。

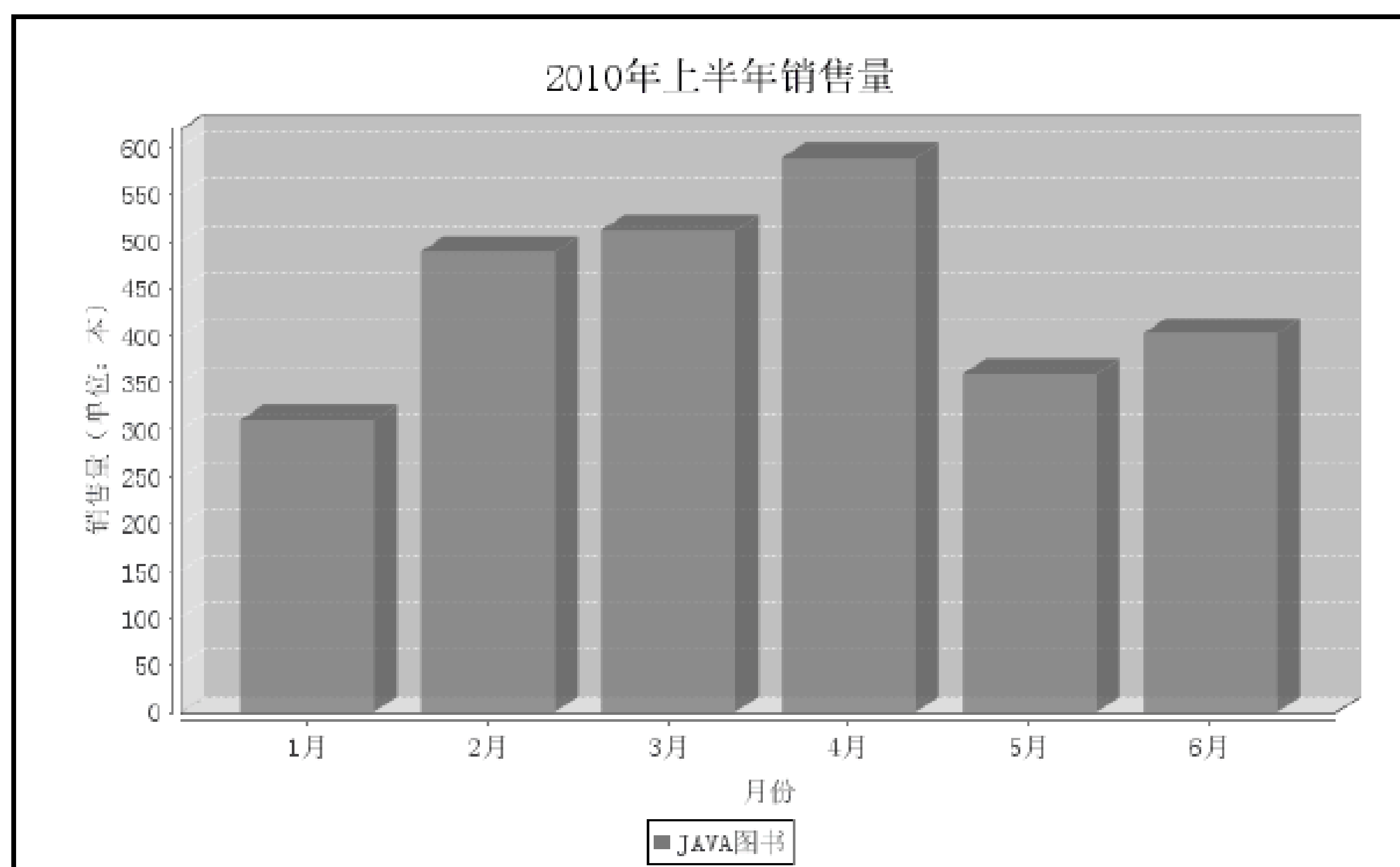


图 8.54　绘制 3D 柱形图

关键技术

在 JFreeChart 中使用 ChartFactory 类的 createBarChart3D()方法可以创建 3D 的柱形图，创建完成后会返回一个 JFreeChart 对象。语法如下：

```
public static JFreeChart createBarChart3D(String title, String categoryAxisLabel, String valueAxisLabel, CategoryDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：3D 柱形图的标题。

❷ categoryAxisLabel：3D 柱形图类别标签，即 X 轴的名称。

❸ valueAxisLabel：3D 柱形图数据标签，即 Y 轴的名称。

❹ dataset：柱形图的数据集合。

❺ orientation：3D 柱形图的图表方向。

❻ legend：表示是否使用图示。

❼ tooltips：表示是否生成工具栏提示。

❽ urls：表示是否生成 URL 链接。

（1）创建 getCategoryDataset()方法，在该方法内部创建一个分类数据集合。代码如下：

```
private CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset(
                    "JAVA 图书", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建 3D 柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart3D("2010 年上半年销售量",        //图表标题
                    "月份",                                                        //X 轴标签
                    "销售量（单位：本）",                                          //Y 轴标签
                    dataset,                                                      //数据集
```

```
                    PlotOrientation.VERTICAL,                         //图表方向：水平、垂直
                    true,                                             //是否显示图例（对于简单的柱形图必须是 false）
                    false,                                            //是否生成工具栏提示
                    false                                             //是否生成 URL 链接
                    );
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中重新设置图表标题、标签等字体。代码如下：

```
public static void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 229：3D 柱形图。

3D 柱形图与普通的柱形图一般的属性都是一样的，例如，设置图表字体、修改图表颜色等。但也有部分属性不能对两种柱形图都起作用，例如，普通柱形图可以设置阴影效果，而 3D 柱形图则没有阴影效果。

实例 230　标记柱形图区间

光盘位置：光盘\MR\08\230

高级

实用指数：★★★★



有时在柱形图显示的范围内，需要划出一部分区域来标注一些特殊内容。本实例将使用特殊颜色显示被划出的区域，标注超出历史最高销售量情况，如图 8.55 所示。



图 8.55　显示区域标记

关键技术

（1）使用 CategoryPlot 类的 addRangeMarker()方法可以创建一个范围标记区域。语法如下：

```
public void addRangeMarker(Marker marker)
```

参数说明

marker：表示范围标记区域。

（2）IntervalMarker 类继承了抽象类 Marker，使用其构造方法可以进行实例化。语法如下：

```
public IntervalMarker(double start, double end)
```

参数说明

❶ start：表示标记区域的开始值。

❷ end：表示标记区域的结束值。

设计过程

（1）创建 getCategoryDataset()方法，在该方法内部创建一个分类数据集合。代码如下：

```
private CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("JAVA 图书", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",    //图表标题
                        "月份",                                                    //X 轴标签
                        "销售量（单位：本）",                                      //Y 轴标签
                        dataset,                                                   //数据集
                        PlotOrientation.VERTICAL,                                  //图表方向：水平、垂直
                        true,                                                      //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                                     //是否生成工具栏提示
                        false                                                      //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中创建一个范围区域，设置范围区域的各种属性。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        CategoryPlot categoryPlot = chart.getCategoryPlot();                    //图表
           IntervalMarker target = new IntervalMarker(560.0, 700.0);
           target.setLabel("超出历史最高销售量");                                //Marker 标签名称
           target.setLabelFont(new Font("宋体", Font.PLAIN, 14));               //Marker 标签字体
           target.setLabelAnchor(RectangleAnchor.LEFT);                         //Marker 标签锚点
           target.setLabelTextAnchor(TextAnchor.BASELINE_LEFT);                 //Marker 标签文字锚点
           target.setPaint(new Color(222, 122, 255, 128));                      //Marker 背景色
           categoryPlot.addRangeMarker(target);                                 //标记范围
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

心法领悟 230：CategoryPlot 类的 addRangeMarker()方法。

使用 CategoryPlot 类的 addRangeMarker()方法，不仅可以在图表中创建多个范围标记区域，还支持层的使用。语法如下：

```
addRangeMarker(Marker marker, Layer layer)
```

参数说明

❶ marker：表示范围标记区域。

❷ layer：表示层的使用情况，Layer 的类中有两个常量，Layer.BACKGROUND 表示区域范围显示在柱形下面一层，Layer.BACKGROUND 表示区域范围显示在柱形上面一层。

实例 231　多系列柱形图　高级

光盘位置：光盘\MR\08\231　实用指数：★★★★★

实例说明

单系列的柱形图只能比较单一分类之间的差异，而使用多系列的柱形图可比较多个分类中的差异以及各个分类之间的不同。本实例将演示如何绘制多系列柱形图，运行结果如图 8.56 所示。

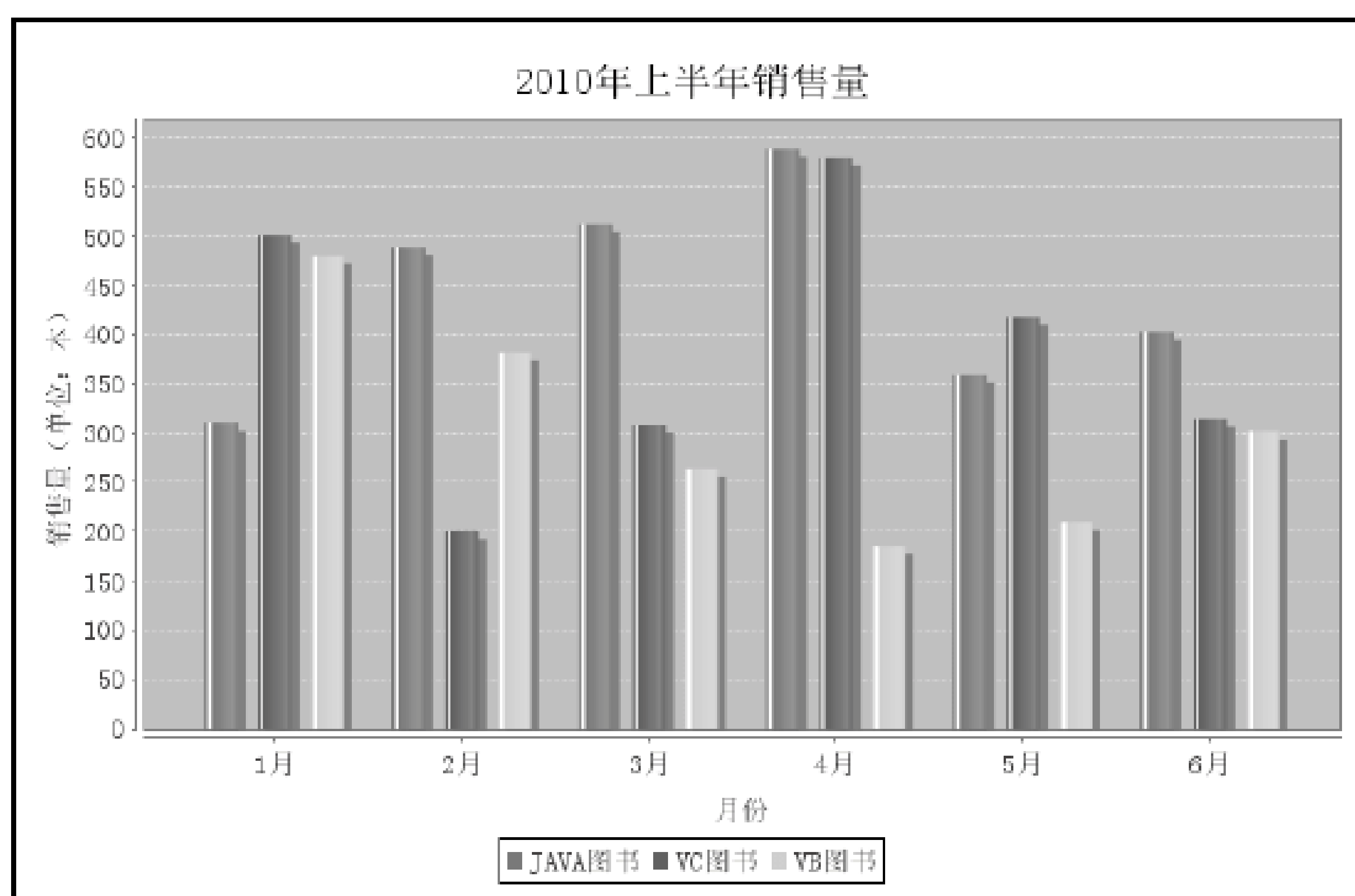


图 8.56　多系列柱形图

关键技术

此处将图表中不同的分类称为系列，而把 X 轴上的刻度称为分类。要绘制多系列的柱形图，首先数据集要能支持多系列的数据。使用 DefaultCategoryDataset 类的 addValue()方法可以添加多系列的数据集。语法如下：

```
public void addValue(double value, Comparable rowKey, Comparable columnKey)
```

参数说明

❶ value：表示柱形图某一系列中某一分类的具体数据。

❷ rowKey：表示柱形图中的系列名称。

❸ columnKey：表示柱形图中的分类名称，也就是 X 轴上的刻度。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建多系列的分类数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //行关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
```

```
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);

        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
    CategoryDataset dataset = getCategoryDataset();
    JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",  //图表标题
                "月份",                                                   //X 轴标签
                "销售量（单位：本）",                                     //Y 轴标签
                dataset,                                                  //数据集
                PlotOrientation.VERTICAL,                                 //图表方向：水平、垂直
                true,                                                     //是否显示图例（对于简单的柱形图必须是 false）
                false,                                                    //是否生成工具栏提示
                false                                                     //是否生成 URL 链接
                );
    return chart;
}
```

（3）创建 updateFont()方法，在该方法中重新设置图表标题、标签等字体。代码如下：

```
public void updateFont(JFreeChart chart) {
    //标题
    TextTitle textTitle = chart.getTitle();
    textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
    //图表（柱形图）
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
    //X 轴字体
    categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
    //X 轴标签字体
    categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
    ValueAxis valueAxis = categoryPlot.getRangeAxis();
    //Y 轴字体
    valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
    //Y 轴标签字体
    valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

心法领悟 231：应用 DatasetUtilities 的 createCategoryDataset()方法。

创建 JFreeChart 数据集的方法有很多，如使用 DatasetUtilities 的 createCategoryDataset()方法可以创建一个多系列的柱形图数据集。语法如下：

```
createCategoryDataset(String rowKeyPrefix, String columnKeyPrefix, double[][] data)
```

参数说明

❶ rowKeyPrefix：表示柱形图中的系列名称。

❷ columnKeyPrefix：表示柱形图中的分类名称，也就是 X 轴上的刻度。

❸ data：表示具体的数据集合。

实例 232　多系列 3D 柱形图　　高级

光盘位置：光盘\MR\08\232　　实用指数：★★★★★

实例说明

多系列的柱形图也可以实现 3D 效果。本实例将演示如何绘制一个多系列的 3D 柱形图，运行结果如图 8.57 所示。

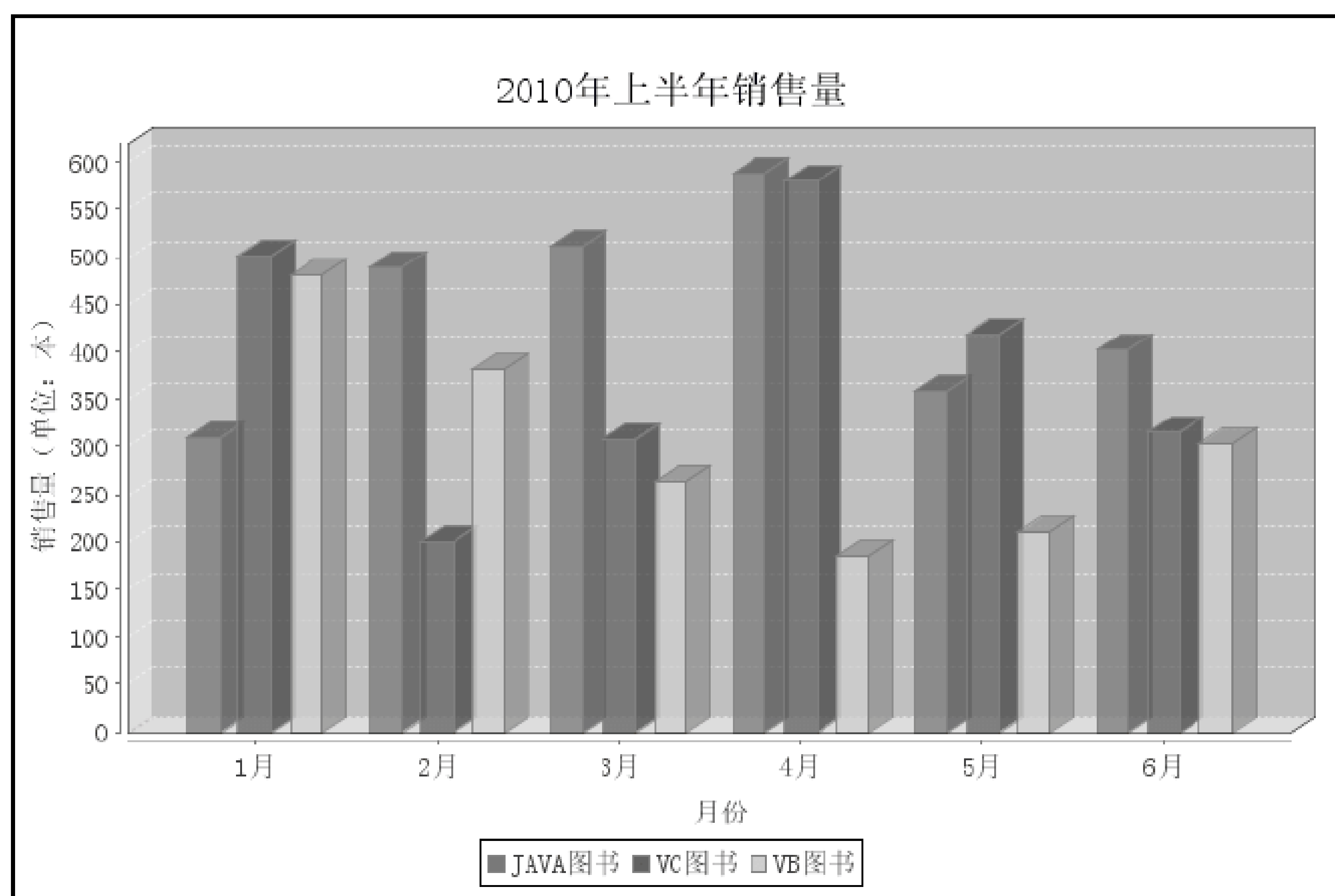


图 8.57　多系列 3D 柱形图

关键技术

在绘制多系列的 3D 柱形图时，还可以设置渲染效果。使用 CategoryPlot 类的 getRenderer()方法可以获取多系列 3D 柱形图的渲染效果实例。语法如下：

```
public CategoryItemRenderer getRenderer()
```

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，在该方法内部创建多系列的分类数据集合。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //行关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
```

```
            final String category6 = "6 月";
            //创建分类数据集
            final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
            dataset.addValue(310, series1, category1);
            dataset.addValue(489, series1, category2);
            dataset.addValue(512, series1, category3);
            dataset.addValue(589, series1, category4);
            dataset.addValue(359, series1, category5);
            dataset.addValue(402, series1, category6);
            dataset.addValue(501, series2, category1);
            dataset.addValue(200, series2, category2);
            dataset.addValue(308, series2, category3);
            dataset.addValue(580, series2, category4);
            dataset.addValue(418, series2, category5);
            dataset.addValue(315, series2, category6);
            dataset.addValue(480, series3, category1);
            dataset.addValue(381, series3, category2);
            dataset.addValue(264, series3, category3);
            dataset.addValue(185, series3, category4);
            dataset.addValue(209, series3, category5);
            dataset.addValue(302, series3, category6);
            return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart3D("2010 年上半年销售量",     //图表标题
                        "月份",                                                    //X 轴标签
                        "销售量（单位：本）",                                        //Y 轴标签
                        dataset,                                                   //数据集
                        PlotOrientation.VERTICAL,                                  //图表方向：水平、垂直
                        true,                                                      //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                                     //是否生成工具栏提示
                        false                                                      //是否生成 URL 链接
                        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 3D 的渲染效果实例，设置柱形图显示边线。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        BarRenderer3D renderer = (BarRenderer3D) categoryPlot.getRenderer();
        //显示边线
        renderer.setDrawBarOutline(true);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

■

心法领悟 232：BarRenderer 类的 setDrawBarOutline()方法。

本实例为多系列的 3D 柱形图设置了边线，设置边线需要使用 BarRenderer 类的 setDrawBarOutline()方法。语法如下：

```
setDrawBarOutline(boolean draw)
```

参数说明

draw：表示是否显示柱形图的边线。当 draw 为 true 时显示边线，为 false 时则不显示边线。

第 9 章

扩展图表技术

- 区域图
- 折线图
- 时序图
- 联合分类图
- 图表的综合应用

9.1 区 域 图

实例 233 基本区域图

光盘位置：光盘\MR\09\233

高级

实用指数：★★★

实例说明

区域图是根据基础数据在图表中绘制一定的区域，通过图表区域的形状和大小体现数据的变化。本实例将演示如何绘制一个基本的区域图，运行结果如图 9.1 所示。

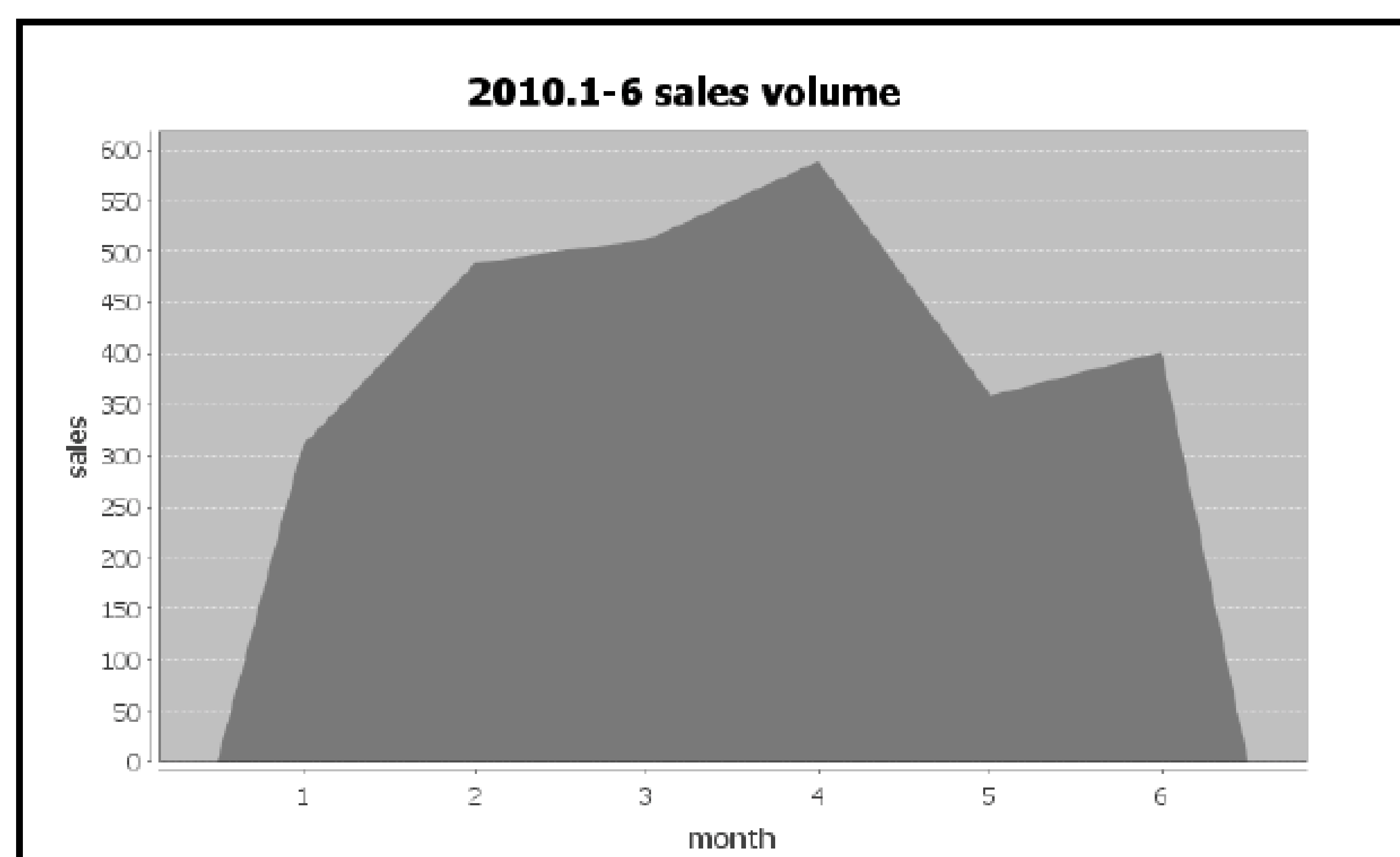


图 9.1 基本区域图

关键技术

ChartFactory 类的 createAreaChart()方法提供了创建基本区域图的方法，创建完成后将返回一个 JFreeChart 对象。语法如下：

```
public static JFreeChart createAreaChart(String title, String categoryAxisLabel, String valueAxisLabel, CategoryDataset dataset, PlotOrientation
        orientation, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：图表的标题。

❷ categoryAxisLabel：图表类别标签，即 X 轴的名称。

❸ valueAxisLabel：图表数据标签，即 Y 轴的名称。

❹ dataset：区域图的数据集合。

❺ orientation：图表的显示方向。

❻ legend：表示是否使用图示。

❼ tooltips：表示是否生成工具栏提示。

❽ urls：表示是否生成 URL 链接。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，向 DefaultKeyedValues 类的实例中添加数据内容，然后通过 DatasetUtilities 类的 createCategoryDataset()方法创建一个数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1", 310);
        keyedValues.addValue("2", 489);
```

```
        keyedValues.addValue("3", 512);
        keyedValues.addValue("4", 589);
        keyedValues.addValue("5", 359);
        keyedValues.addValue("6", 402);
        //创建数据集
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createAreaChart("2010.1-6 sales volume",        //图表标题
        "month",                                                                        //X 轴标签
        "sales",                                                                        //Y 轴标签
        dataset,                                                                        //数据集
        PlotOrientation.VERTICAL,                                                       //图表方向：水平、垂直
        true,                                                                           //是否显示图例
        false,                                                                          //是否生成工具栏提示
        false                                                                           //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码请参见配书光盘。

秘笈心法

心法领悟 233：addValue()方法。

本实例使用 DefaultKeyedValues 类的 addValue()方法向实例中添加数据。语法如下：

```
addValue(Comparable key, double value)
```

参数说明

❶ key：图表的关键字，在本实例中是区域图中的分类。

❷ value：图表类别的值，在本实例中指区域图中分类的数值。

实例 234	显示多分类区域图 光盘位置：光盘\MR\09\234	高级 实用指数：★★★★

实例说明

普通的区域图只能显示一个分类的区域信息，而多分类的区域图可以更清楚地显示出不同分类之间的差别。本实例将演示如何显示一个多分类区域图，运行结果如图 9.2 所示。

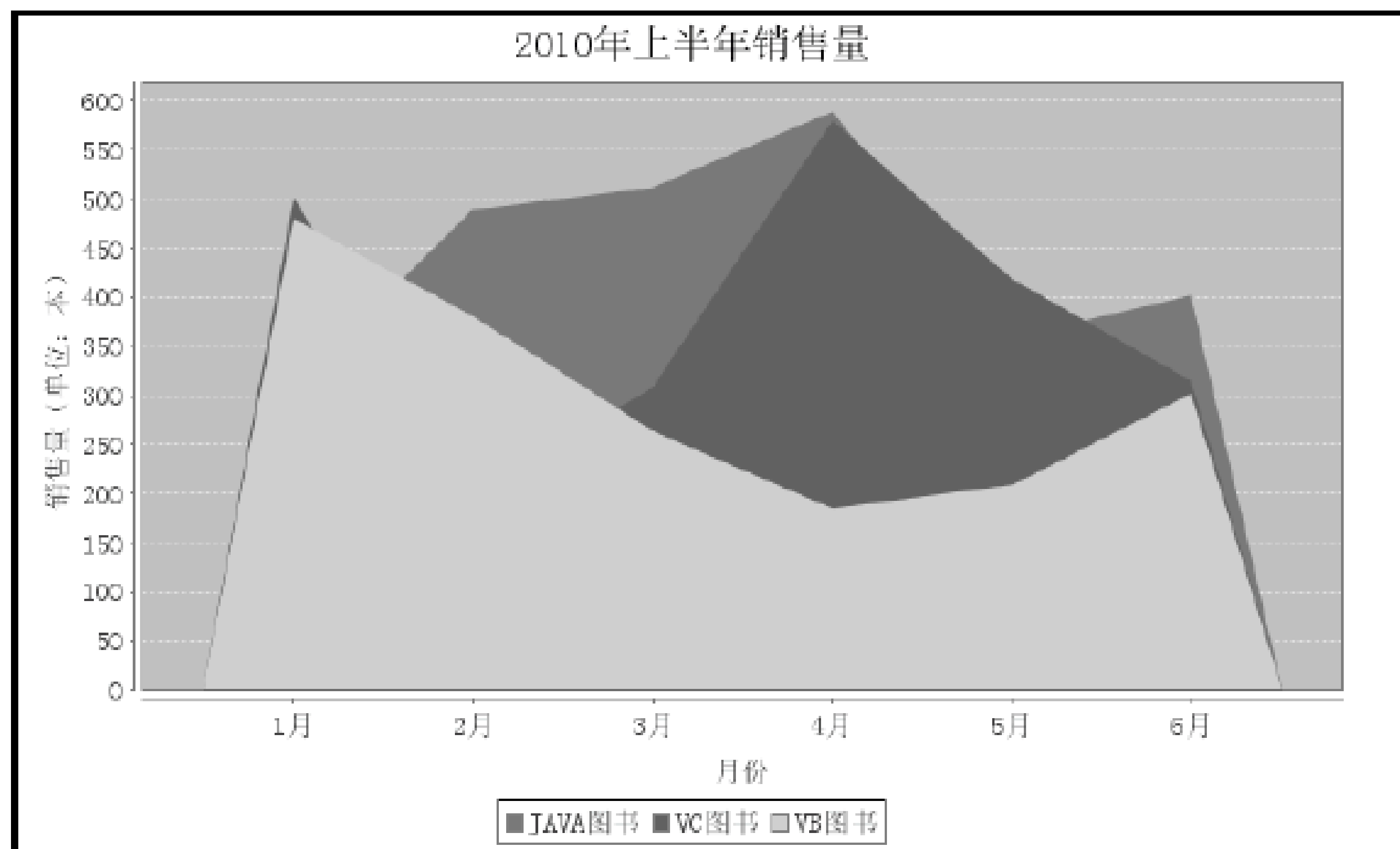


图 9.2　多分类区域图

关键技术

使用 DefaultCategoryDataset 类的 addValue()方法可以添加多类别的数据，为 JFreeChart 提供多类别的数据集合。语法如下：

```
public void addValue(double value, Comparable rowKey, Comparable columnKey)
```

参数说明

❶ value：表示某一分类的数据值。

❷ rowKey：数据集的行关键字，用来表示图表的系列名称。

❸ columnKey：数据集的列关键字，用来表示图表类别名称。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据，生成分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //行关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，再使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createAreaChart("2010 年上半年销售量",        //图表标题
        "月份",                                                                       //X 轴标签
        "销售量（单位：本）",                                                         //Y 轴标签
        dataset,                                                                      //数据集
        PlotOrientation.VERTICAL,                                                     //图表方向：水平、垂直
        true,                                                                         //是否显示图例
        false,                                                                        //是否生成工具栏提示
        false                                                                         //是否生成 URL 链接
        );
```

```
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改图表标题、X 轴、Y 轴、X 轴标签及 Y 轴标签等字体。代码如下：

```
public static void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 234：显示区域的问题。

显示多分类区域时每个系列都形成一个区域，不同的区域显示内容可能会形成交叉，如果某一系列的区域值较小时可能会被其他区域完全覆盖，所以有时如果在图表中找不到某一区域图时，并不是没有显示，而是被较大区域覆盖。

实例 235	设置区域图透明度 光盘位置：光盘\MR\09\235	高级 实用指数：★★★★

实例说明

区域图中某一系列的区域值较小时，可能会被其他区域完全覆盖，使用者无法看到。本实例将对区域图设置一定的透明度，以方便查看图表，运行结果如图 9.3 所示。

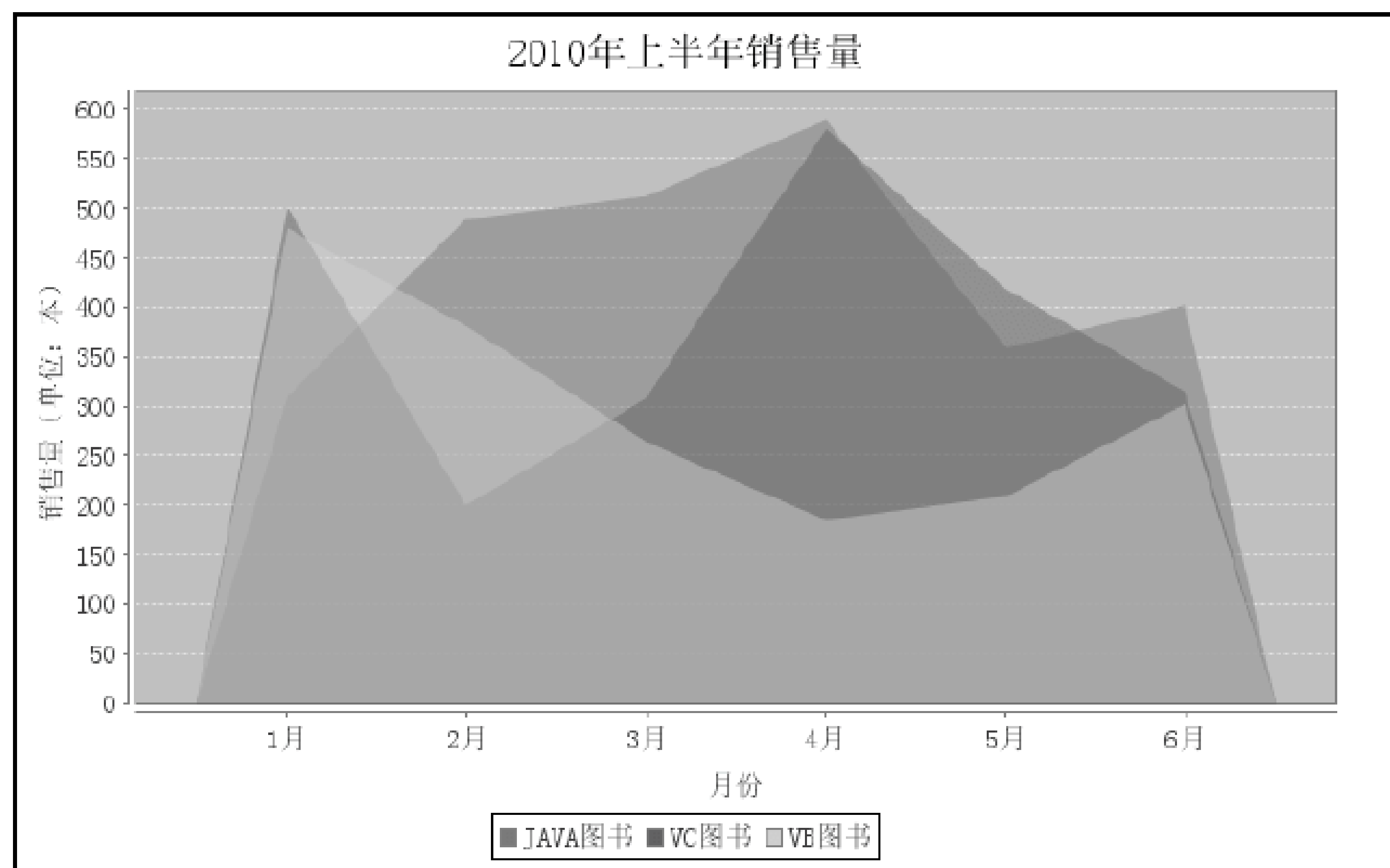


图 9.3　显示透明区域图

关键技术

使用 CategoryPlot 类的 setForegroundAlpha()方法可以设置图表的透明度。语法如下：

```
public void setForegroundAlpha(float alpha)
```

参数说明

alpha：表示图表的透明度，数值范围为 0～1。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据，生成分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
	//行关键字
	final String series1 = "JAVA 图书";
	final String series2 = "VC 图书";
	final String series3 = "VB 图书";
	//列关键字
	final String category1 = "1 月";
	final String category2 = "2 月";
	final String category3 = "3 月";
	final String category4 = "4 月";
	final String category5 = "5 月";
	final String category6 = "6 月";
	//创建分类数据集
	final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
	dataset.addValue(310, series1, category1);
	dataset.addValue(489, series1, category2);
	dataset.addValue(512, series1, category3);
	dataset.addValue(589, series1, category4);
	dataset.addValue(359, series1, category5);
	dataset.addValue(402, series1, category6);
	dataset.addValue(501, series2, category1);
	dataset.addValue(200, series2, category2);
	dataset.addValue(308, series2, category3);
	dataset.addValue(580, series2, category4);
	dataset.addValue(418, series2, category5);
	dataset.addValue(315, series2, category6);
	dataset.addValue(480, series3, category1);
	dataset.addValue(381, series3, category2);
	dataset.addValue(264, series3, category3);
	dataset.addValue(185, series3, category4);
	dataset.addValue(209, series3, category5);
	dataset.addValue(302, series3, category6);
	return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
	CategoryDataset dataset = getCategoryDataset();
	JFreeChart chart = ChartFactory.createAreaChart("2010 年上半年销售量",	//图表标题
	"月份",					//X 轴标签
	"销售量（单位：本）",			//Y 轴标签
	dataset,				//数据集
	PlotOrientation.VERTICAL,		//图表方向：水平、垂直
	true,					//是否显示图例
	false,					//是否生成工具栏提示
	false					//是否生成 URL 链接
	);
	return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 类，然后设置透明度为 0.5f。代码如下：

```
private void updatePlot(JFreeChart chart) {
	//分类图表
```

```
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置透明度
        categoryPlot.setForegroundAlpha(0.5f);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 235：区域图的透明度范围。

区域图的透明度范围为 0～1，默认值为 1。值越小，透明度越高，值越大，透明度越低。当透明度为 0 时，区域图将被隐藏；当透明度为 1 时，区域图将无法透明。

实例 236　添加说明文字

光盘位置：光盘\MR\09\236

高级　实用指数：★★★

实例说明

在区域图中添加说明文字，可以解释图表的含义，还可以表达图表的意义等。本实例将演示如何添加说明文字，运行结果如图 9.4 所示。

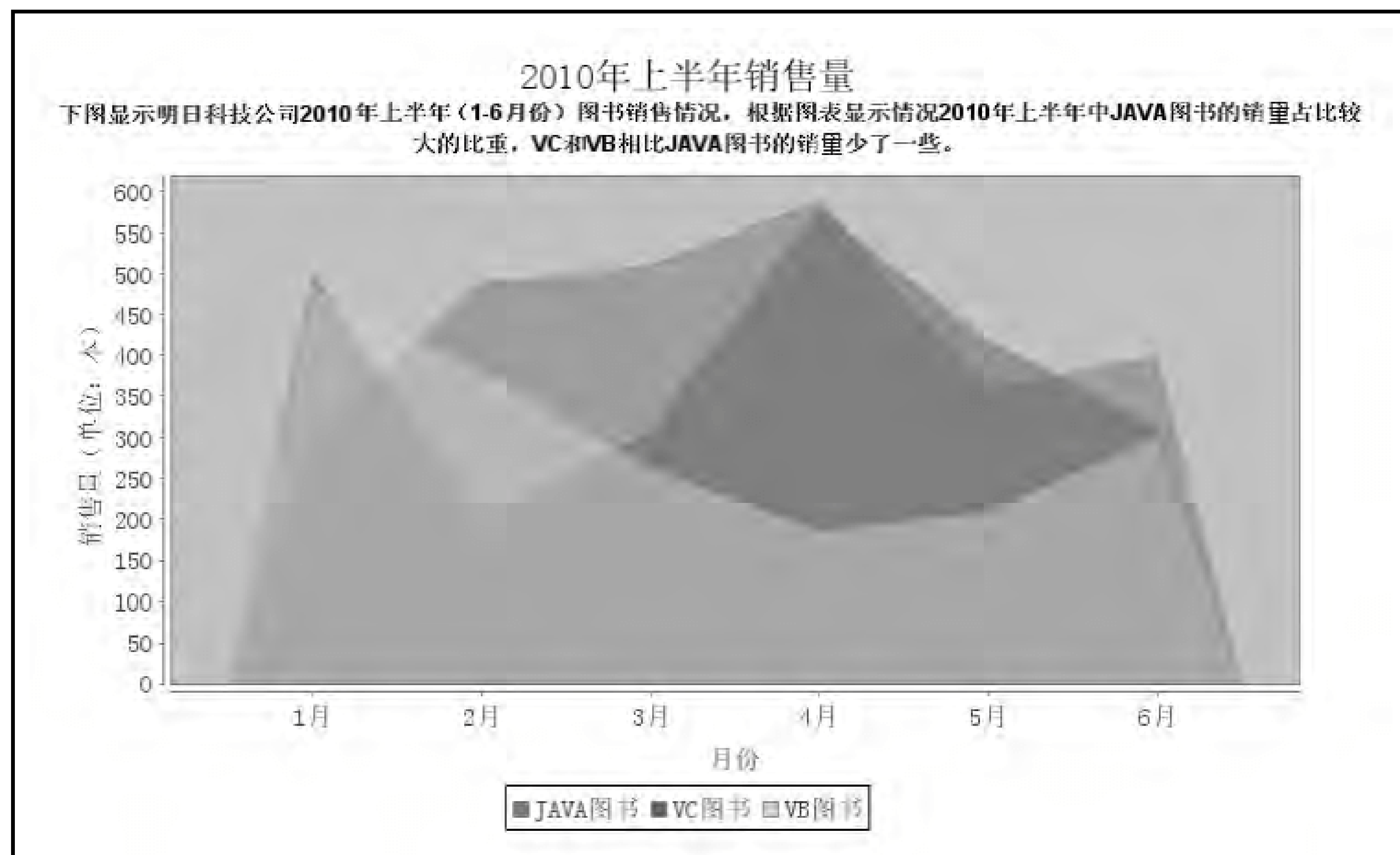


图 9.4　添加说明文字

关键技术

使用 JFreeChart 类的 addSubtitle()方法可以为区域图添加标题，同时还可添加一些说明性的文字。语法如下：

```
public void addSubtitle(Title subtitle)
```

参数说明

subtitle：表示为图表添加 Title 实例。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据，生成分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //行关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
```

```
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）编写 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createAreaChart("2010 年上半年销售量",     //图表标题
        "月份",                                                                 //X 轴标签
        "销售量（单位：本）",                                                   //Y 轴标签
        dataset,                                                                //数据集
        PlotOrientation.VERTICAL,                                               //图表方向：水平、垂直
        true,                                                                   //是否显示图例
        false,                                                                  //是否生成工具栏提示
        false                                                                   //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 类，为图表添加说明文字。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        categoryPlot.setForegroundAlpha(0.5f);
        //添加说明文字
        TextTitle subtitle = new TextTitle("下图显示明日科技公司 2010 上半年（1-6 月份）图书销售情况，根据图表显示情况，2010 年上半年中
            JAVA 图书的销量占比较大的比重，VC 和 VB 相比 JAVA 图书的销量少了一些。");
        chart.addSubtitle(subtitle);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

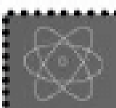
心法领悟 236：TextTitle 类的构造函数。

为区域图添加说明性文字，可以使用 TextTitle。TextTitle 构造函数的语法如下：

```
TextTitle(String text)
```

参数说明

text：表示要添加的文字内容。



实例 237　**设置说明文字位置**　**高级**

光盘位置：光盘\MR\09\237　　实用指数：★★

实例说明

区域图中的说明文字默认情况下都显示在图表上部，字体为黑色。本实例将把区域图说明文字设置成绿色，并且显示在图表左侧，运行结果如图 9.5 所示。

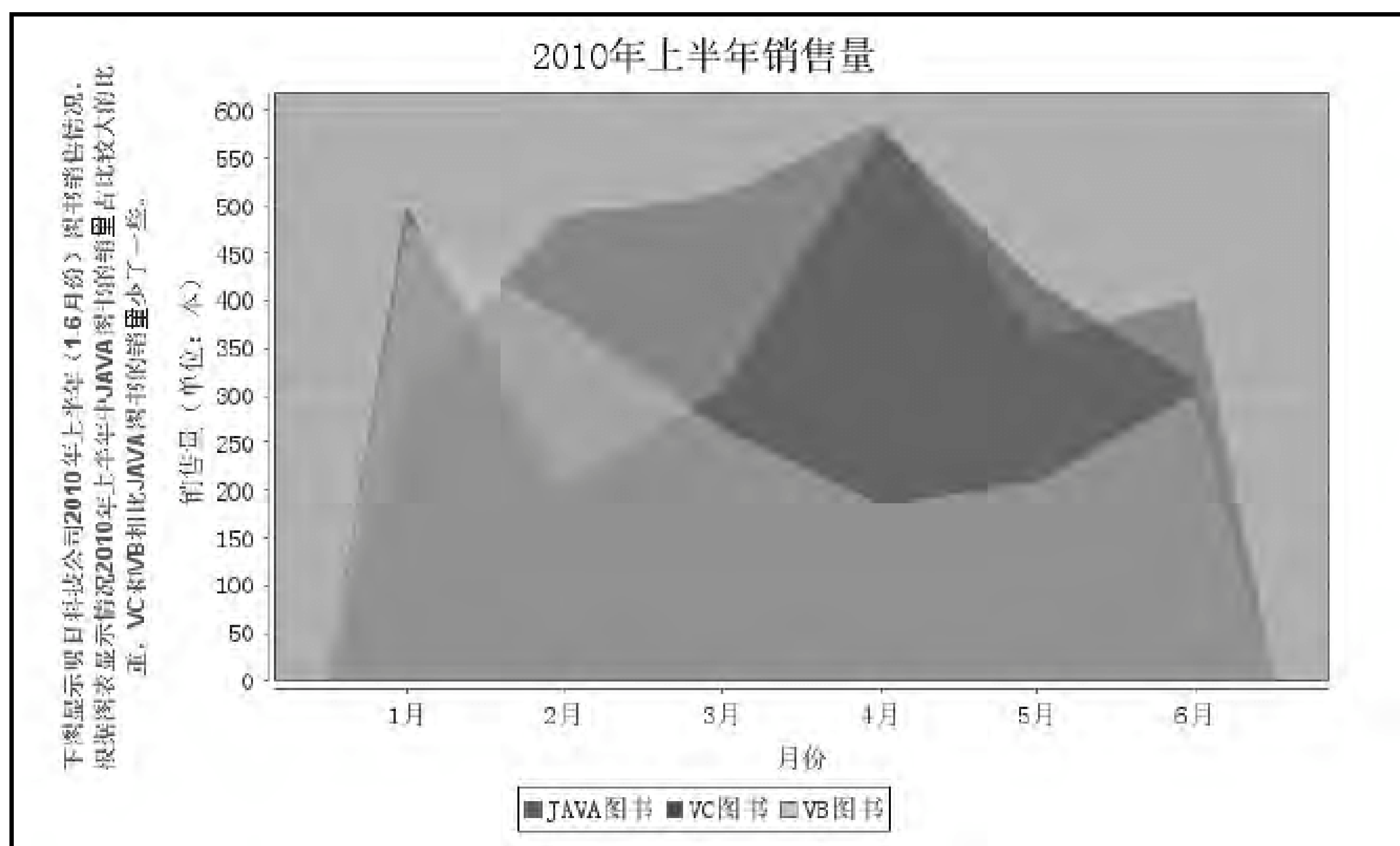


图 9.5　设置说明文字位置

关键技术

使用 Title 类的 setPosition()方法可以为区域图的说明文字设置方位，包括图表上、下、左、右 4 个方位。语法如下：

```
public void setPosition(RectangleEdge position)
```

参数说明

position：表示为图表的说明文字设置方位。

（1）创建 ChartUtil 类，编写 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据，生成分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //行关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
```

```
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createAreaChart("2010 年上半年销售量",     //图表标题
        "月份",                                                                  //X 轴标签
        "销售量（单位：本）",                                                      //Y 轴标签
        dataset,                                                                 //数据集
        PlotOrientation.VERTICAL,                                                //图表方向：水平、垂直
        true,                                                                    //是否显示图例
        false,                                                                   //是否生成工具栏提示
        false                                                                    //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改图表标题、X 轴、Y 轴、X 轴标签及 Y 轴标签等字体。代码如下：

```
private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（4）创建 updatePlot()方法，在该方法中获取 CategoryPlot 类，将图表说明文字置于图表左侧，同时设置文字颜色为绿色。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        categoryPlot.setForegroundAlpha(0.5f);
        TextTitle subtitle = new TextTitle("下图显示明日科技公司 2010 年上半年（1-6 月份）图书销售情况，根据图表显示情况，2010 年上半年
            中 JAVA 图书的销量占比较大的比重，VC 和 VB 相比 JAVA 图书的销量少了一些。");
        //设置显示位置
        subtitle.setPosition(RectangleEdge.LEFT);
        subtitle.setPaint(Color.GREEN);
        chart.addSubtitle(subtitle);
}
```

（5）创建 index.jsp 页面，显示 JFreeChart 生成的图表，具体代码参见配书光盘。

秘笈心法

心法领悟 237：使用 RectangleEdge 设置图表说明文字的方位。

RectangleEdge 中有 4 个常量用于表示说明文字的 4 个位置，RectangleEdge.TOP 表示位于图表上部，RectangleEdge.BOTTOM 表示位于图表底部，RectangleEdge.LEFT 表示位于图表左侧，RectangleEdge.RIGHT 表示位于图表右侧。



实例 238　**设置区域图 X 轴显示位置**　高级

光盘位置：光盘\MR\09\238　实用指数：★★★

实例说明

区域图的 X 坐标轴默认显示在图表底部，本实例将演示如何将其显示在图表顶部，运行结果如图 9.6 所示。

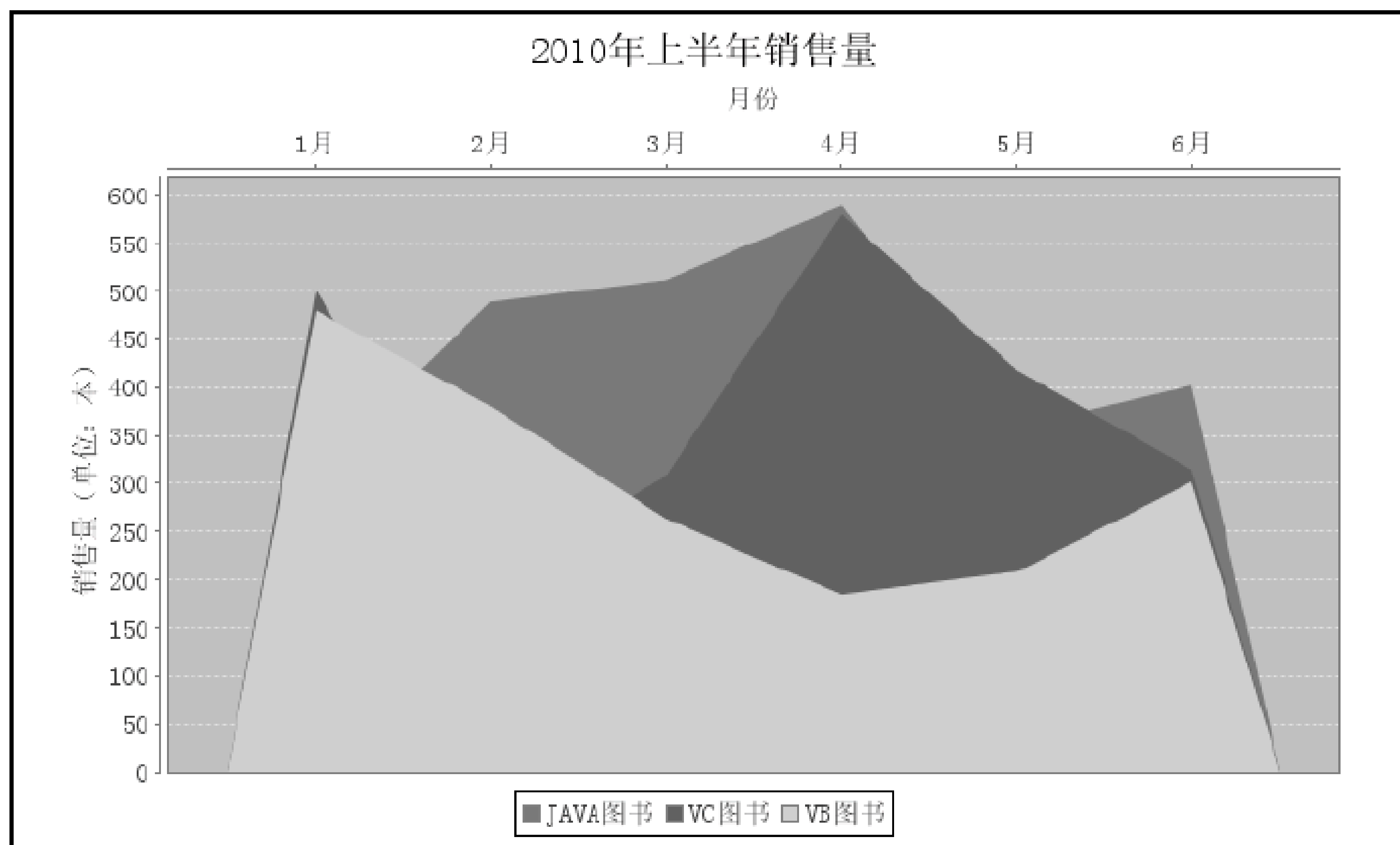


图 9.6　X 轴显示在图表顶部

关键技术

使用 CategoryPlot 类的 setDomainAxisLocation()方法可以设置区域图 X 轴的位置。语法如下：

```
public void setDomainAxisLocation(AxisLocation location)
```

参数说明

location：表示区域图 X 轴的方位。使用参数 AxisLocation.TOP_OR_LEFT，表示设置 X 轴显示在图表顶部或左侧；使用参数 AxisLocation.BOTTOM_OR_LEFT，表示设置 X 轴显示在图表底部或左侧。

设计过程

（1）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据，生成分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //行关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
```

```
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);

        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createAreaChart("2010 年上半年销售量",    //图表标题
        "月份",                                                                //X 轴标签
        "销售量（单位：本）",                                                  //Y 轴标签
        dataset,                                                               //数据集
        PlotOrientation.VERTICAL,                                              //图表方向：水平、垂直
        true,                                                                  //是否显示图例
        false,                                                                 //是否生成工具栏提示
        false                                                                  //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 类，为图表 X 轴设置显示位置，将其显示在图表顶部。代码如下：

```
private void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //区域图 X 轴显示位置
        categoryPlot.setDomainAxisLocation(AxisLocation.TOP_OR_LEFT);
}
```

（4）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

■

心法领悟 238：设置区域图 Y 轴的位置。

使用 CategoryPlot 类的 setRangeAxisLocation()方法可以设置区域图 Y 轴的位置。语法如下：

```
setRangeAxisLocation(AxisLocation location)
```

参数说明

location：表示区域图 Y 轴的方位。使用参数 AxisLocation.TOP_OR_RIGHT，表示设置 Y 轴显示在图表顶部或右侧；使用参数 AxisLocation.BOTTOM_OR_RIGHT，表示设置 Y 轴显示在图表底部或右侧。



实例 239　设置区域图 X 轴标签角度

光盘位置：光盘\MR\09\239

高级

实用指数：★★★

实例说明

可以根据需要对区域图的坐标轴标签进行角度调整。本实例将演示如何调整区域图 X 轴的标签角度，运行结果如图 9.7 所示。

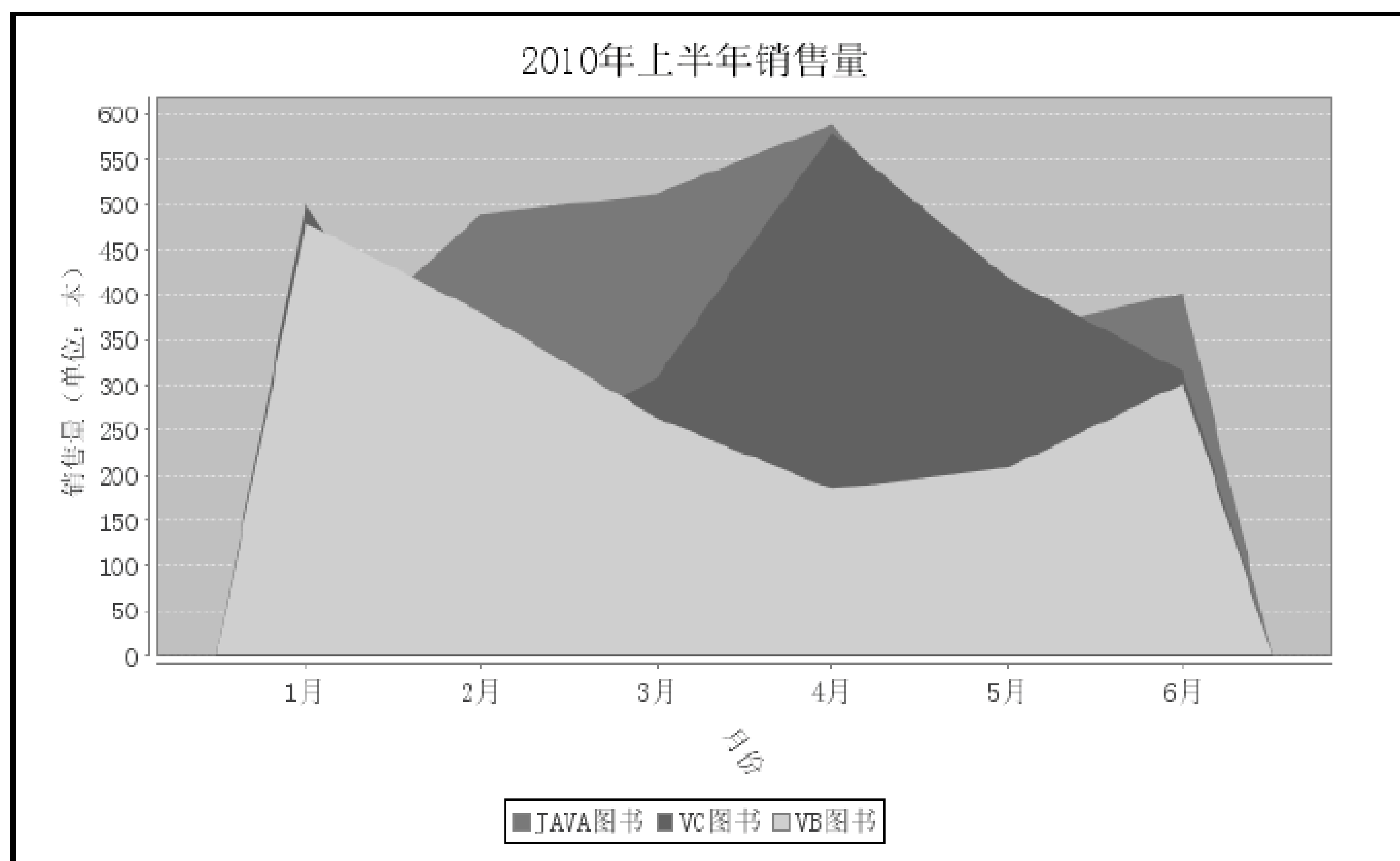


图 9.7　设置 X 轴标签角度

关键技术

使用 CategoryAxis 类的 setLabelAngle()方法可以设置区域图 X 轴标签的角度。语法如下：

```
public void setLabelAngle(double angle)
```

参数说明

angle：表示区域图 X 轴的标签旋转角度，此角度根据弧度计算。

（1）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据，生成分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //行关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
```

```
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createAreaChart("2010 年上半年销售量",          //图表标题
        "月份",                                                                      //X 轴标签
        "销售量（单位：本）",                                                        //Y 轴标签
        dataset,                                                                     //数据集
        PlotOrientation.VERTICAL,                                                    //图表方向：水平、垂直
        true,                                                                        //是否显示图例
        false,                                                                       //是否生成工具栏提示
        false                                                                        //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改图表标题、X 轴、Y 轴、X 轴标签及 Y 轴标签等字体。代码如下：

```
public static void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（4）创建 updatePlot()方法，在该方法中获取 CategoryPlot 类，将区域图 X 轴标签弧度设置为 Math.PI*0.3。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //区域图 X 轴标签角度
        categoryAxis.setLabelAngle(Math.PI*0.3);
}
```

心法领悟 239：设置区域图 Y 轴标签的角度。

区域图 Y 轴标签的角度可以根据需要自由设置。ValueAxis 类的 setLabelAngle()方法用于设置区域图 Y 轴标签的角度。语法如下：

```
setLabelAngle(double angle)
```

参数说明

angle：表示区域图 Y 轴标签的旋转角度，此角度根据弧度进行计算。

实例 240　设置区域图 X 轴尺度标签角度

光盘位置：光盘\MR\09\240

高级

实用指数：★★★

实例说明

区域图坐标轴的尺度标签可以自由调整。本实例将演示如何调整区域图 X 轴尺度标签的角度，运行结果如图 9.8 所示。

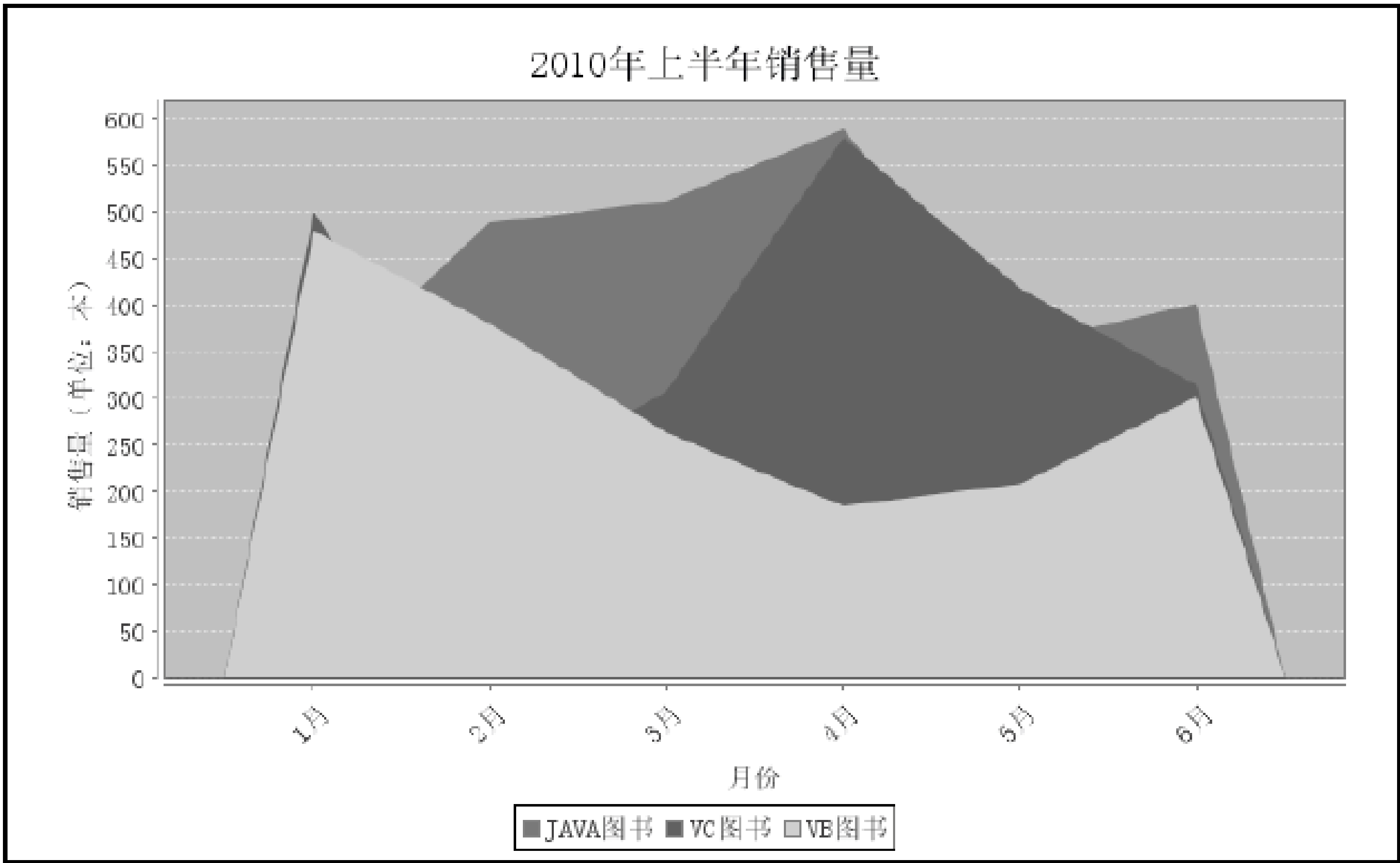


图 9.8　调整 X 轴尺度标签角度

关键技术

使用 CategoryAxis 类的 setCategoryLabelPositions()方法可以设置区域图 X 轴尺度标签的角度。语法如下：

```
public void setCategoryLabelPositions(CategoryLabelPositions positions)
```

参数说明

positions：表示区域图 X 轴尺度标签的旋转角度，使用 CategoryLabelPositions 类中定义的常量进行设置。

###

（1）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据，生成分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //行关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
```

```
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，再使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createAreaChart("2010 年上半年销售量",         //图表标题
        "月份",                                                                   //X 轴标签
        "销售量（单位：本）",                                                       //Y 轴标签
        dataset,                                                                 //数据集
        PlotOrientation.VERTICAL,                                                //图表方向：水平、垂直
        true,                                                                    //是否显示图例
        false,                                                                   //是否生成工具栏提示
        false                                                                    //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在方法中获取 CategoryPlot 类，为图表 X 轴尺度标签设置角度为向上 45°，代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //区域图 X 轴尺度标签角度
        categoryAxis.setCategoryLabelPositions(CategoryLabelPositions.UP_45);
}
```

秘笈心法

心法领悟 240：区域图 Y 轴尺度标签的角度。

可以使用 ValueAxis 类的 setVerticalTickLabels()方法设置区域图 Y 轴尺度标签的角度。语法如下：

```
setVerticalTickLabels(boolean flag)
```

参数说明

flag：表示区域图 Y 轴的尺度标签是否垂直。



实例 241 设置区域颜色

光盘位置：光盘\MR\09\241

高级

实用指数：★★★★★

区域图的颜色可以根据需要进行修改。本实例将演示如何修改区域图指定系列的颜色，运行结果如图 9.9 所示。

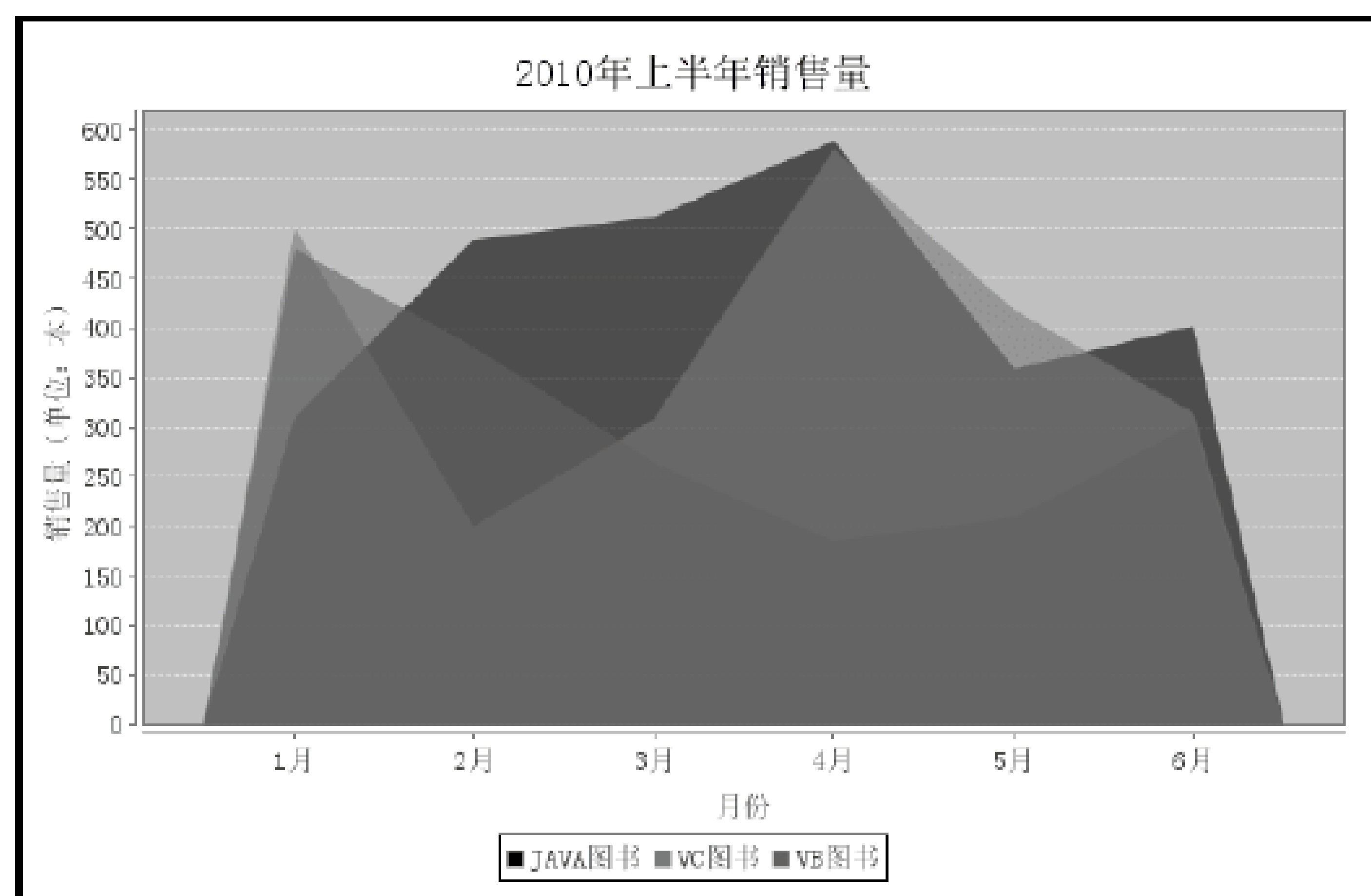


图 9.9　设置区域颜色

关键技术

使用 AreaRenderer 类的 setSeriesPaint()方法可以设置区域图指定系列的颜色。语法如下：

```
public void setSeriesPaint(int series, Paint paint)
```

参数说明

❶ series：表示指定区域图的系列。

❷ paint：表示指定区域图的颜色。

（1）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据，生成分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //行关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createAreaChart()方

法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createAreaChart("2010 年上半年销售量",      //图表标题
        "月份",                                                               //X 轴标签
        "销售量（单位：本）",                                                  //Y 轴标签
        dataset,                                                              //数据集
        PlotOrientation.VERTICAL,                                             //图表方向：水平、垂直
        true,                                                                 //是否显示图例
        false,                                                                //是否生成工具栏提示
        false                                                                 //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 类，然后通过该类获取 AreaRenderer 实例，将区域图指定系列设置为黑色。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置透明度
        categoryPlot.setForegroundAlpha(0.6f);
        AreaRenderer renderer = (AreaRenderer) categoryPlot.getRenderer();
        //设置区域颜色
        renderer.setSeriesPaint(0, Color.BLACK);
}
```

秘笈心法

心法领悟 241：设置区域图系列颜色时需要注意的问题。

系列颜色与背景颜色不能一致，否则用户在查看图表时无法区分区域图中的颜色。在设置图表系列颜色的同时，JFreeChart 会让图示与图表中的颜色自动保持一致。

9.2 折 线 图

实例 242 创建基本折线图

光盘位置：光盘\MR\09\242

高级

实用指数：★★★★★

实例说明

折线图通过数据的线形走势来体现情况的变化趋势，其中 X 轴表示分类情况，Y 轴表示具体数值。本实例将演示如何创建一个基本的折线图，运行结果如图 9.10 所示。

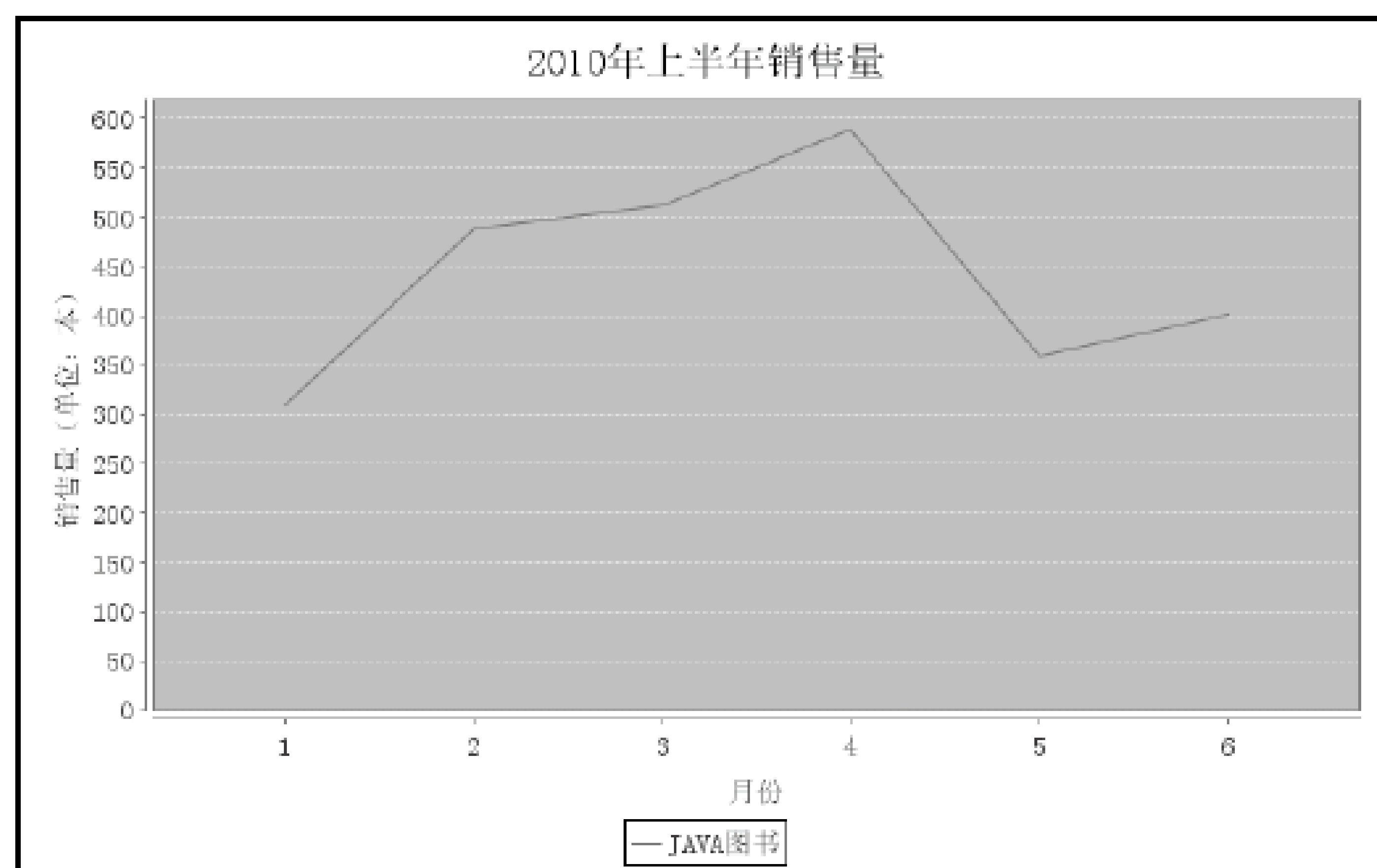


图 9.10　基本折线图

■ 关键技术

ChartFactory 类的 createLineChart()方法用来创建基本折线图，创建完成后将返回一个 JFreeChart 对象。语法如下：

```
createLineChart (String title, String categoryAxisLabel, String valueAxisLabel, CategoryDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：图表的标题。

❷ categoryAxisLabel：图表类别标签，即 X 轴的名称。

❸ valueAxisLabel：图表数据标签，即 Y 轴的名称。

❹ dataset：折线图的数据集合。

❺ orientation：图表的显示方向。

❻ legend：表示是否使用图示。

❼ tooltips：表示是否生成工具栏提示。

❽ urls：表示是否生成 URL 链接。

■

（1）创建 getCategoryDataset()方法，向 DefaultKeyedValues 类的实例中添加数据内容，然后通过 DatasetUtilities 类的 createCategoryDataset()方法创建一个数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        keyedValues.addValue("1", 310);
        keyedValues.addValue("2", 489);
        keyedValues.addValue("3", 512);
        keyedValues.addValue("4", 589);
        keyedValues.addValue("5", 359);
        keyedValues.addValue("6", 402);
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("JAVA 图书", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createLineChart("2010 年上半年销售量",          //图表标题
                "月份",                                                                //X 轴标签
                "销售量（单位：本）",                                                  //Y 轴标签
                dataset,                                                               //数据集
                PlotOrientation.VERTICAL,                                              //图表方向：水平、垂直
                true,                                                                  //是否显示图例
                false,                                                                 //是否生成工具栏提示
                false                                                                  //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改标题、X 轴、Y 轴、X 轴标签及 Y 轴标签等字体。代码如下：

```
public static void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
```

```
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

秘笈心法

心法领悟 242：设置 X 轴和 Y 轴字体。

折线图的 X 轴、Y 轴字体使用不同的类进行设置，X 轴字体使用 CategoryAxis 类设置，Y 轴字体使用 ValueAxis 类设置；二者都可以使用 setLabelFont()方法修改标签字体。

实例 243　创建多条折线图

光盘位置：光盘\MR\09\243

高级

实用指数：★★★★★

实例说明

折线图支持多系列图表，通过折线的走势与不同折线的比较体现情况变化趋势。本实例将演示如何创建多条折线图，运行结果如图 9.11 所示。

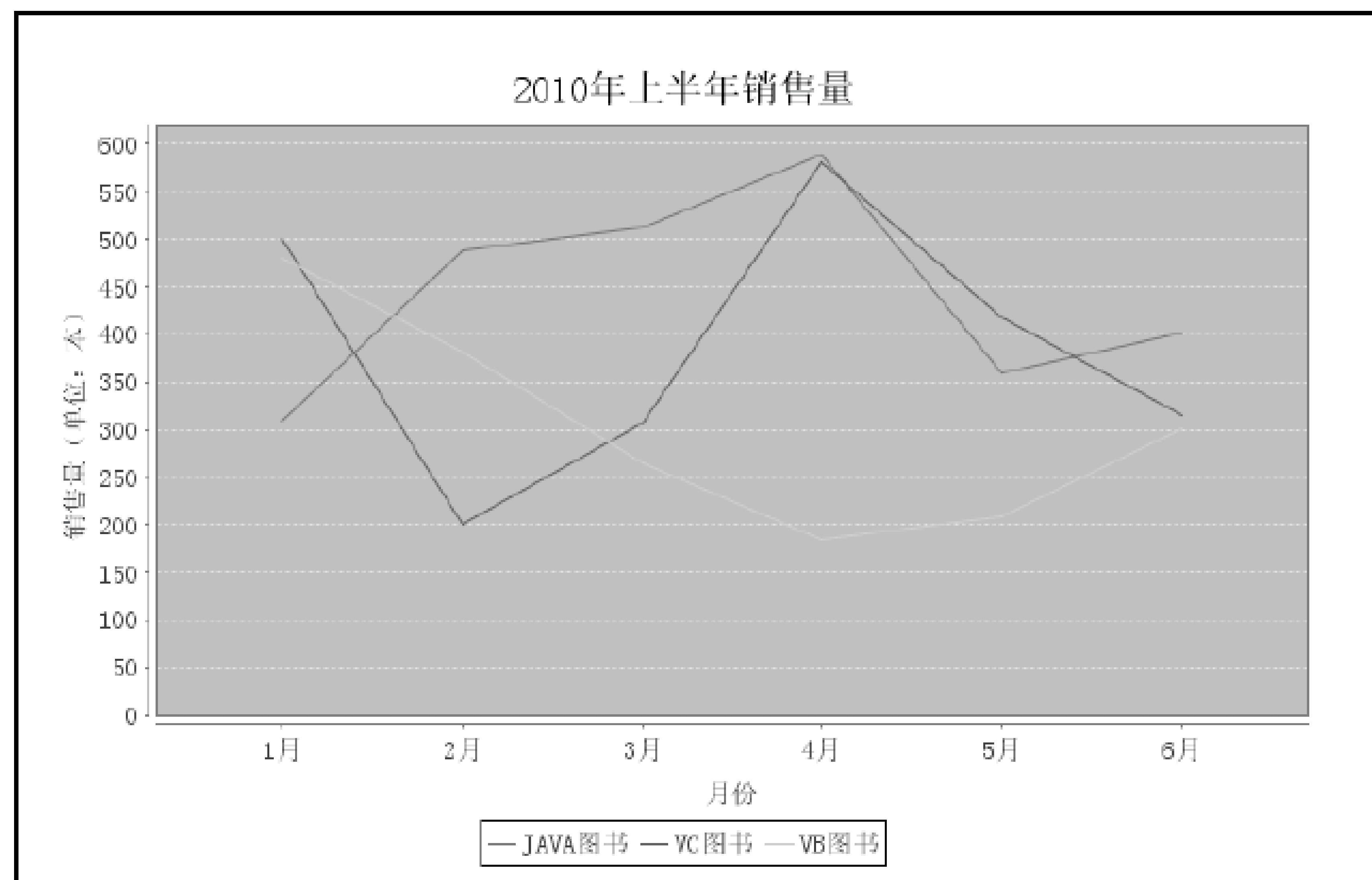


图 9.11　多条折线图

关键技术

利用 DefaultCategoryDataset 类的 addValue()方法可以创建多系列数据集，使用多系列数据集即可创建多条折线图。语法如下：

```
addValue(double value, Comparable rowKey, Comparable columnKey)
```

参数说明

❶ value：表示具体数据值。

❷ rowKey：表示 X 轴系列名称。

❸ columnKey：表示 X 轴分类名称。

（1）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，创建分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
    //系列关键字
    final String series1 = "JAVA 图书";
    final String series2 = "VC 图书";
    final String series3 = "VB 图书";
    //分类关键字
    final String category1 = "1 月";
    final String category2 = "2 月";
    final String category3 = "3 月";
    final String category4 = "4 月";
    final String category5 = "5 月";
    final String category6 = "6 月";
    //创建分类数据集
    final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
    dataset.addValue(310, series1, category1);
    dataset.addValue(489, series1, category2);
    dataset.addValue(512, series1, category3);
    dataset.addValue(589, series1, category4);
    dataset.addValue(359, series1, category5);
    dataset.addValue(402, series1, category6);
    dataset.addValue(501, series2, category1);
    dataset.addValue(200, series2, category2);
    dataset.addValue(308, series2, category3);
    dataset.addValue(580, series2, category4);
    dataset.addValue(418, series2, category5);
    dataset.addValue(315, series2, category6);
    dataset.addValue(480, series3, category1);
    dataset.addValue(381, series3, category2);
    dataset.addValue(264, series3, category3);
    dataset.addValue(185, series3, category4);
    dataset.addValue(209, series3, category5);
    dataset.addValue(302, series3, category6);
    return dataset;
}
```

（2）创建 getJFreeChart()方法，在方法中获取数据集合，然后使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
    CategoryDataset dataset = getCategoryDataset();
    JFreeChart chart = ChartFactory.createLineChart("2010 年上半年销售量",     //图表标题
        "月份",                                                               //X 轴标签
        "销售量（单位：本）",                                                 //Y 轴标签
        dataset,                                                              //数据集
        PlotOrientation.VERTICAL,                                             //图表方向：水平、垂直
        true,                                                                 //是否显示图例
        false,                                                                //是否生成工具栏提示
        false                                                                 //是否生成 URL 链接
    );
    return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改标题、X 轴、Y 轴、X 轴标签及 Y 轴标签等字体。代码如下：

```
public static void updateFont(JFreeChart chart) {
    //标题
    TextTitle textTitle = chart.getTitle();
    textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
    LegendTitle legendTitle = chart.getLegend();
    legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
    //图表
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
    //X 轴字体
    categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
    //X 轴标签字体
    categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
    ValueAxis valueAxis = categoryPlot.getRangeAxis();
    //Y 轴字体
```

```
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

秘笈心法

心法领悟 243：创建多系列数据集。

DefaultKeyedValues 类只能创建基本线形图，因为在该类中无法添加多系列数据；而使用 DefaultCategoryDataset 类则可以创建多折线图，因为它可以创建多系列数据集。



实例 244 创建水平折线图

光盘位置：光盘\MR\09\244

高级

实用指数：★★★★

实例说明

折线图可以绘制成垂直样式或水平样式，水平样式的折线图 X 轴显示在图表左侧，Y 轴显示在图表上部。本例将演示如何创建水平折线图，运行结果如图 9.12 所示。

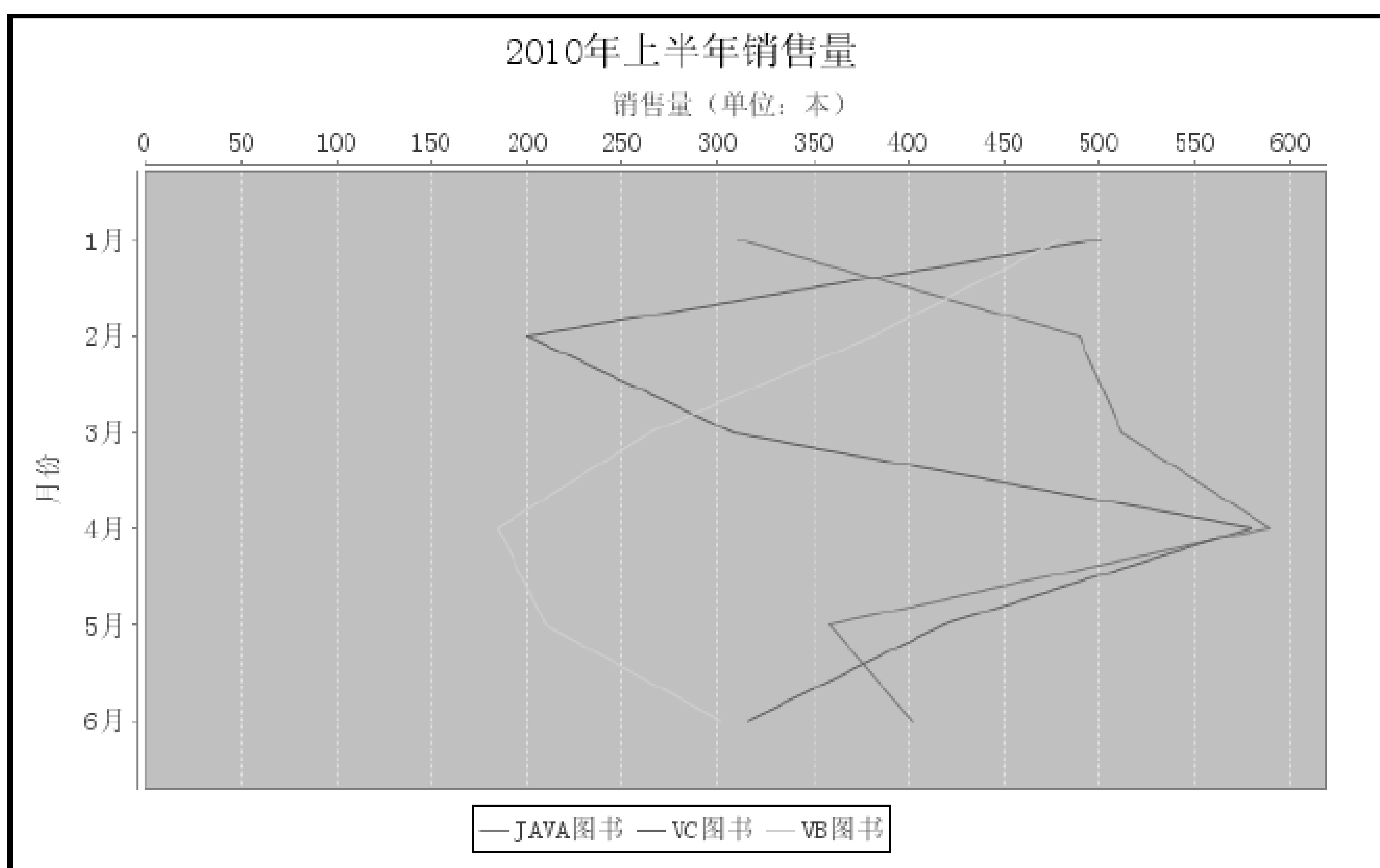


图 9.12　水平折线图

关键技术

通过 PlotOrientation 类可以设置折线图的显示状态。PlotOrientation 类有两个常量，PlotOrientation.HORIZONTAL 表示折线图显示为水平状态，PlotOrientation.VERTICAL 表示折线图显示为垂直状态。

（1）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，创建分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //系列关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //分类关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
```

```
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createLineChart("2010 年上半年销售量",        //图表标题
                "月份",                                                        //X 轴标签
                "销售量（单位：本）",                                            //Y 轴标签
                dataset,                                                       //数据集
                PlotOrientation.VERTICAL,                                      //图表方向：水平、垂直
                true,                                                          //是否显示图例
                false,                                                         //是否生成工具栏提示
                false                                                          //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改标题、X 轴、Y 轴、X 轴标签及 Y 轴标签等字体。代码如下：

```
private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

心法领悟 244：设置折线图 X 轴的位置。

使用 CategoryPlot 类的 setDomainAxisLocation()方法可以设置折线图 X 轴的位置。语法如下：

```
setDomainAxisLocation(AxisLocation location)
```

参数说明

location：表示折线图 X 轴的方位。使用参数 AxisLocation.TOP_OR_RIGHT，表示设置 X 轴显示在图表顶部或右侧；使用参数 AxisLocation.BOTTOM_OR_RIGHT，表示设置 X 轴显示在图表底部或右侧。



实例 245　隐藏折线图中指定系列的折线

光盘位置：光盘\MR\09\245

高级

实用指数：★★★★★

实例说明

绘制多条折线图时，在数据集中可能有一部分系列不需要显示在图表中，此时可将其隐藏。本实例将演示如何隐藏指定折线图中指定系列的折线，运行结果如图 9.13 所示。

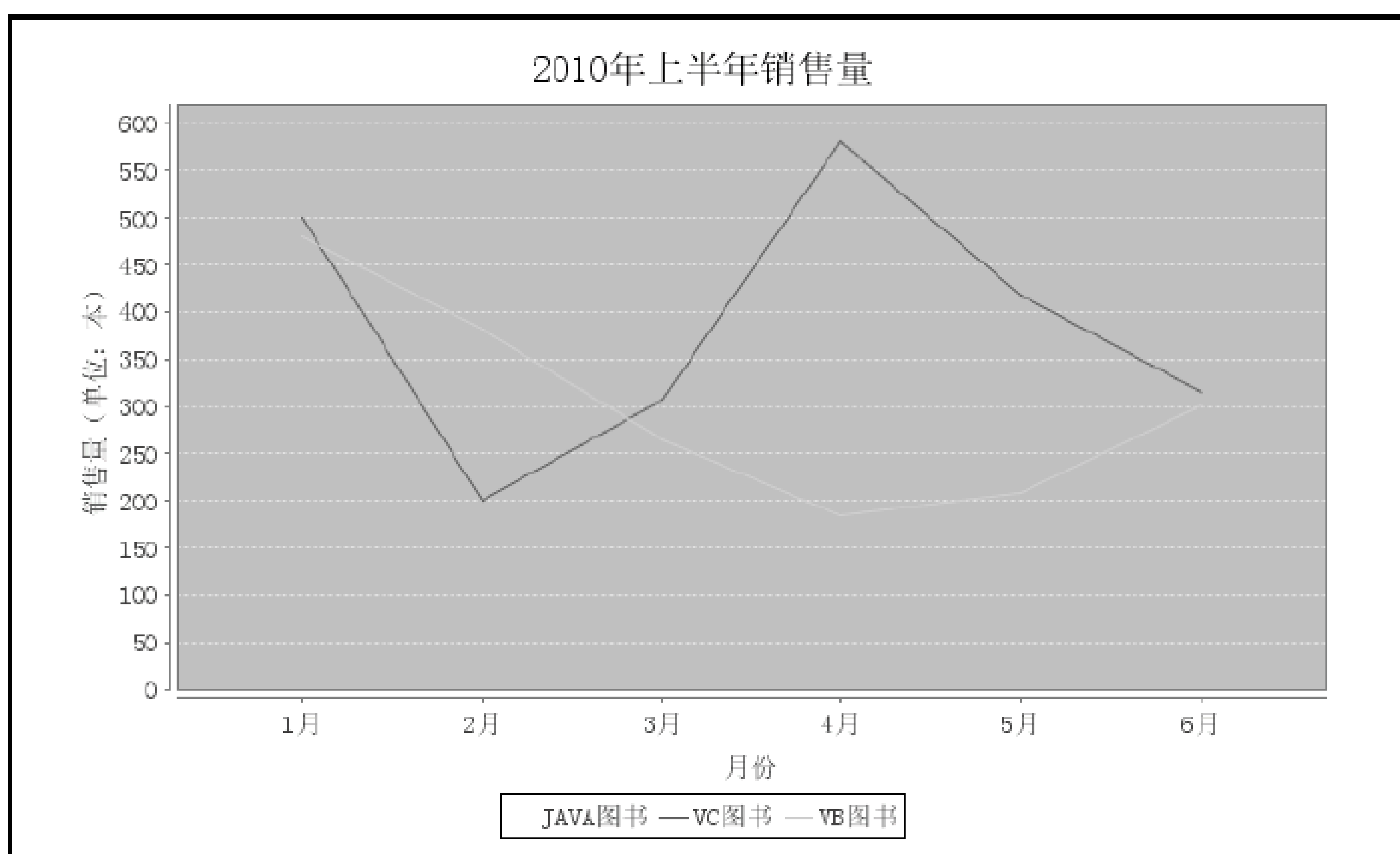


图 9.13　隐藏指定系列的折线

关键技术

使用 LineAndShapeRenderer 类的 setSeriesLinesVisible()方法可以隐藏折线图中指定系列的折线。语法如下：

```
setSeriesLinesVisible(int series, boolean visible)
```

参数说明

❶ series：表示折线图中的系列。

❷ visible：表示折线图中指定系列是否显示，当 visible 为 true 时显示折线；为 false 时则隐藏折线。

（1）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，创建分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //系列关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //分类关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
```

```
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createLineChart("2010 年上半年销售量",          //图表标题
                "月份",                                                      //X 轴标签
                "销售量（单位：本）",                                          //Y 轴标签
                dataset,                                                     //数据集
                PlotOrientation.VERTICAL,                                    //图表方向：水平、垂直
                true,                                                        //是否显示图例
                false,                                                       //是否生成工具栏提示
                false                                                        //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 类，然后根据该类获取 LineAndShapeRenderer 实例，隐藏第一个系列的折线。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        LineAndShapeRenderer renderer = (LineAndShapeRenderer) categoryPlot.getRenderer();
        //隐藏指定折线
        renderer.setSeriesLinesVisible(0, false);
}
```

秘笈心法

心法领悟 245：隐藏折线图。

绘制折线图时，如果使用 setSeriesLinesVisible()方法隐藏折线图，折线将不会在图表中显示，同时当前系列的图示在图表中也会被隐藏。

实例 246	加粗折线 光盘位置：光盘\MR\09\246	高级 实用指数：★★★★

实例说明

折线图中的折线笔触默认情况下比较细，用户可以根据需要加粗折线的笔触。本实例将演示如何加粗折线，运行结果如图 9.14 所示。

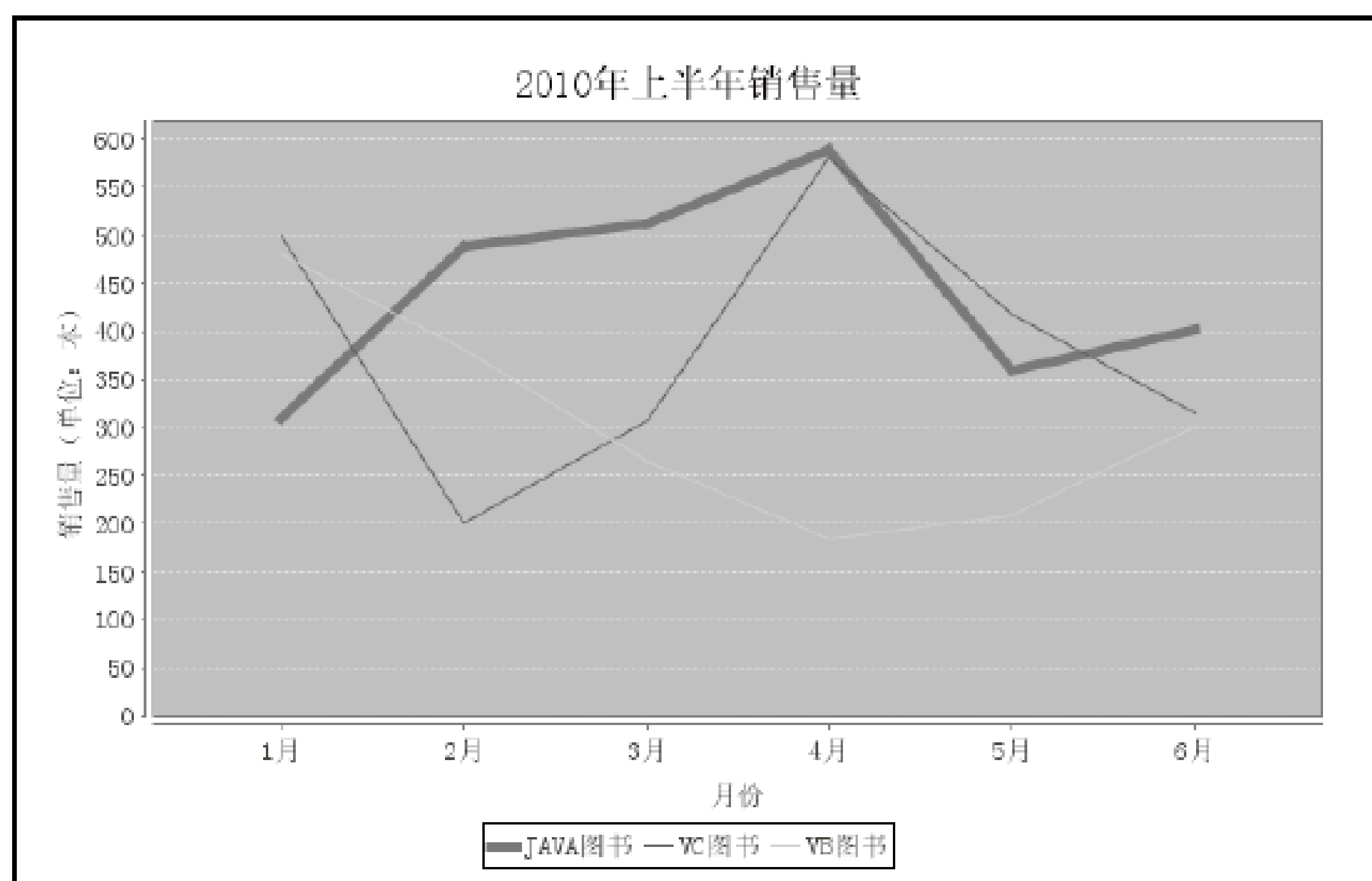


图 9.14　加粗折线

关键技术

通过修改折线图的笔触可以重新绘制折线图。使用 LineAndShapeRenderer 类的 setSeriesStroke()方法可以设置折线图的笔触。语法如下：

```
setSeriesStroke(int series, Stroke stroke)
```

参数说明

❶ series：表示折线图中的系列。

❷ stroke：表示需要修改的折线图的笔触。

（1）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，创建分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //系列关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //分类关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
	CategoryDataset dataset = getCategoryDataset();
	JFreeChart chart = ChartFactory.createLineChart("2010 年上半年销售量",		//图表标题
		"月份",									//X 轴标签
		"销售量（单位：本）",							//Y 轴标签
		dataset,									//数据集
		PlotOrientation.VERTICAL,						//图表方向：水平、垂直
		true,										//是否显示图例
		false,										//是否生成工具栏提示
		false										//是否生成 URL 链接
	);
	return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 CategoryPlot 类，然后根据该类获取 LineAndShapeRenderer 实例，加粗第一个系列的折线。代码如下：

```
public static void updatePlot(JFreeChart chart) {
	//分类图表
	CategoryPlot categoryPlot = chart.getCategoryPlot();
	LineAndShapeRenderer renderer = (LineAndShapeRenderer) categoryPlot.getRenderer();
	//加粗线形
	renderer.setSeriesStroke(0,new BasicStroke(5));
}
```

秘笈心法

心法领悟 246：设置折线图笔触。

使用 BasicStroke 类的构造方法，可以生成折线图笔触实例，通过修改这个笔触可以修改折线的形状。语法如下：

```
BasicStroke(float width)
```

参数说明

width：表示折线图中笔触的粗细。

实例 247	显示折线节点 光盘位置：光盘\MR\09\247	高级 实用指数：★★★★

在绘制折线图时，可以为折线生成节点，以突出显示关键数据。本实例将演示如何显示折线节点，运行结果如图 9.15 所示。

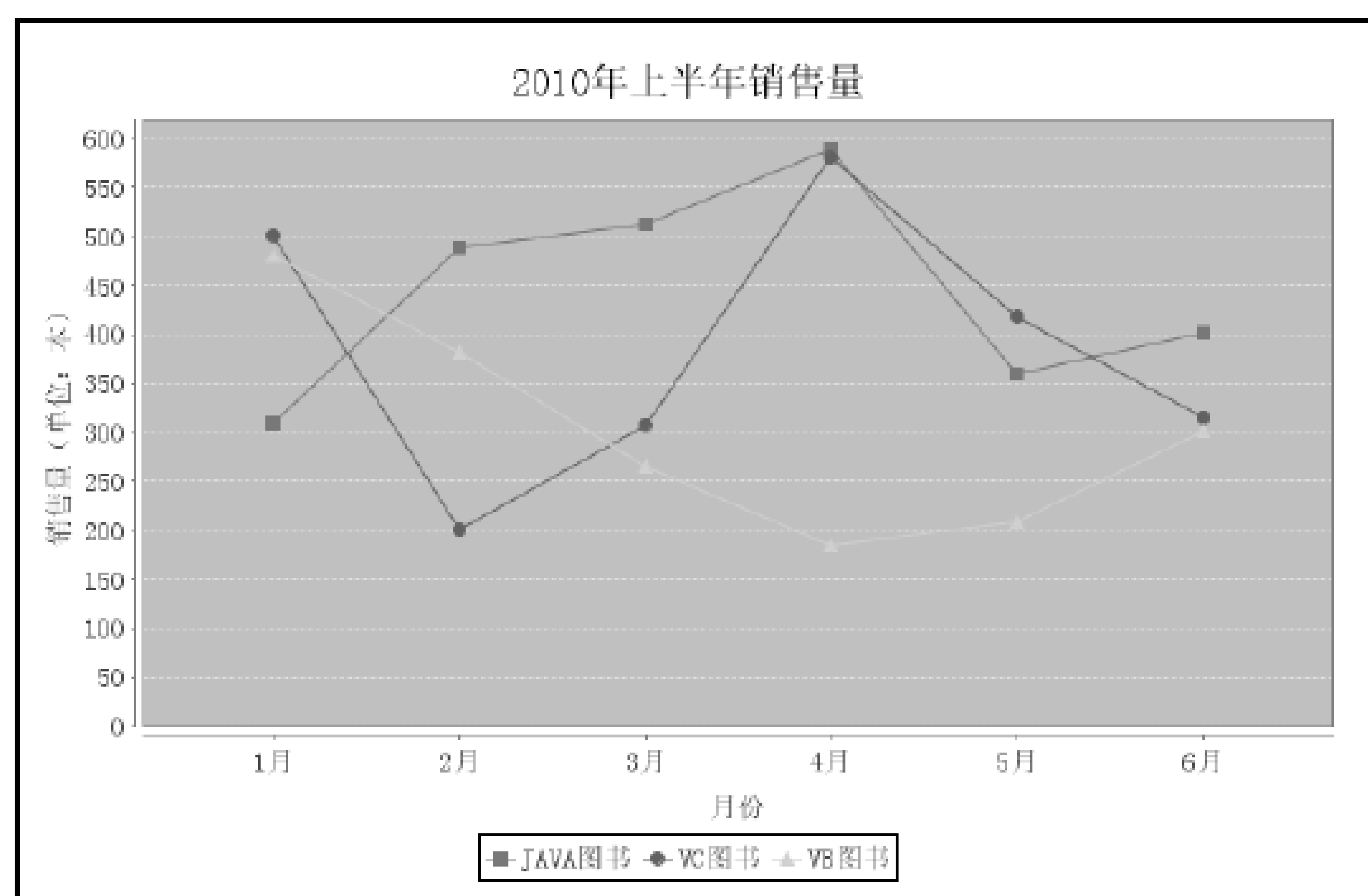


图 9.15　显示折线节点

关键技术

使用 LineAndShapeRenderer 类的 setSeriesShapesVisible()方法可以设置是否显示折线图的节点。语法如下：

```
setSeriesShapesVisible(int series, boolean visible)
```

参数说明

❶ series：表示折线图中的系列。

❷ visible：表示是否显示节点，当 visible 为 true 时显示节点，当 visible 为 false 时则不显示节点。

（1）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，创建分类数据集，代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //系列关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //分类关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createLineChart("2010 年上半年销售量",          //图表标题
                "月份",                                                                  //X 轴标签
                "销售量（单位：本）",                                                    //Y 轴标签
                dataset,                                                                 //数据集
                PlotOrientation.VERTICAL,                                                //图表方向：水平、垂直
                true,                                                                    //是否显示图例
                false,                                                                   //是否生成工具栏提示
                false                                                                    //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 LineAndShapeRenderer 实例，通过该实例显示折线图节点。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        LineAndShapeRenderer renderer = (LineAndShapeRenderer) categoryPlot.getRenderer();
        //显示节点
        renderer.setSeriesShapesVisible(0, true);
        renderer.setSeriesShapesVisible(1, true);
        renderer.setSeriesShapesVisible(2, true);
}
```

秘笈心法

心法领悟 247：折线图的节点。

显示折线图的节点时，每条折线不但颜色不一致，其节点样式也不同，这是为了在查看折线图时可以更明显地区分各条折线之间的节点。

实例 248　生成节点图

光盘位置：光盘\MR\09\248

高级

实用指数：★★★★

实例说明

折线图中折线以及节点的显示与隐藏可以根据需要自由控制。本实例将把折线隐藏起来，同时显示折线节点，这时图表中只显示出节点之间的对比情况，效果如图 9.16 所示。

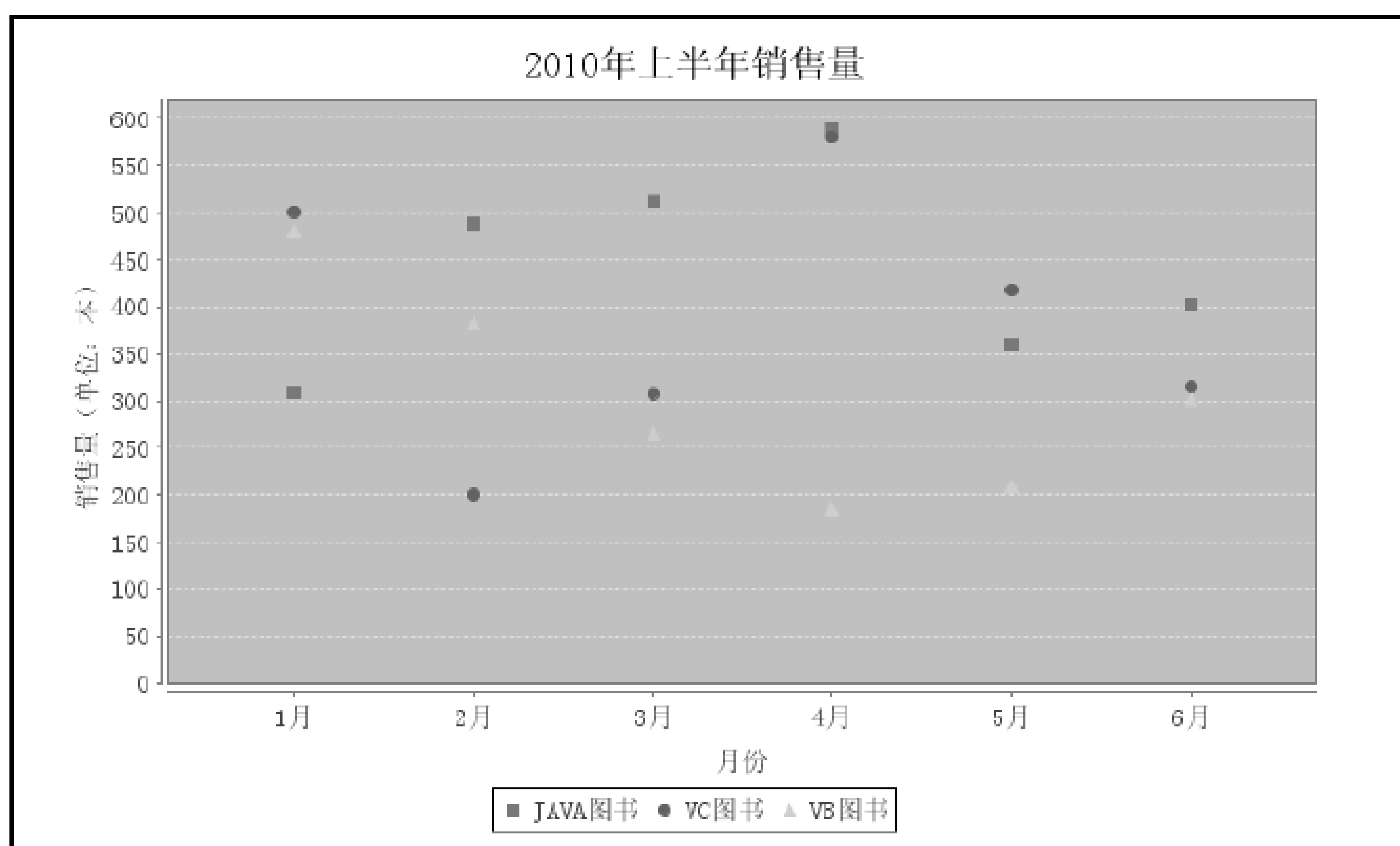


图 9.16　只显示节点

关键技术

使用 LineAndShapeRenderer 类的 setSeriesLinesVisible()方法可以设置是否显示折线图中的折线。语法如下：

```
setSeriesLinesVisible(int series, boolean visible)
```

参数说明

❶ series：表示折线图中的系列。

❷ visible：表示是否显示折线，当 visible 为 true 时显示折线，为 false 时则不显示折线。

（1）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，创建分类数据集。

代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //系列关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //分类关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createLineChart("2010 年上半年销售量",        //图表标题
                "月份",                                                            //X 轴标签
                "销售量（单位：本）",                                                //Y 轴标签
                dataset,                                                           //数据集
                PlotOrientation.VERTICAL,                                          //图表方向：水平、垂直
                true,                                                              //是否显示图例
                false,                                                             //是否生成工具栏提示
                false                                                              //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 LineAndShapeRenderer 实例，通过该实例隐藏折线，然后显示折线的节点。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        LineAndShapeRenderer renderer = (LineAndShapeRenderer) categoryPlot.getRenderer();
        //显示节点
        renderer.setSeriesLinesVisible(0, false);
        renderer.setSeriesLinesVisible(1, false);
        renderer.setSeriesLinesVisible(2, false);
        renderer.setSeriesShapesVisible(0, true);
        renderer.setSeriesShapesVisible(1, true);
        renderer.setSeriesShapesVisible(2, true);
}
```

秘笈心法

心法领悟 248：只显示折线图的节点。

本实例使用 LineAndShapeRenderer 类的 setSeriesLinesVisible()方法把折线图中的全部折线隐藏起来，同时使用 LineAndShapeRenderer 类的 setSeriesShapesVisible()方法显示出折线图中所有的节点，这样图表中就只能展示各个节点之间的对比情况了。

实例 249　绘制虚线折线图

光盘位置：光盘\MR\09\249

高级

实用指数：★★★★

实例说明

折线图默认的笔触是实线，用户可以根据实际需要设置不同的折线图笔触。本实例将演示如何绘制虚线折线图，运行结果如图 9.17 所示。

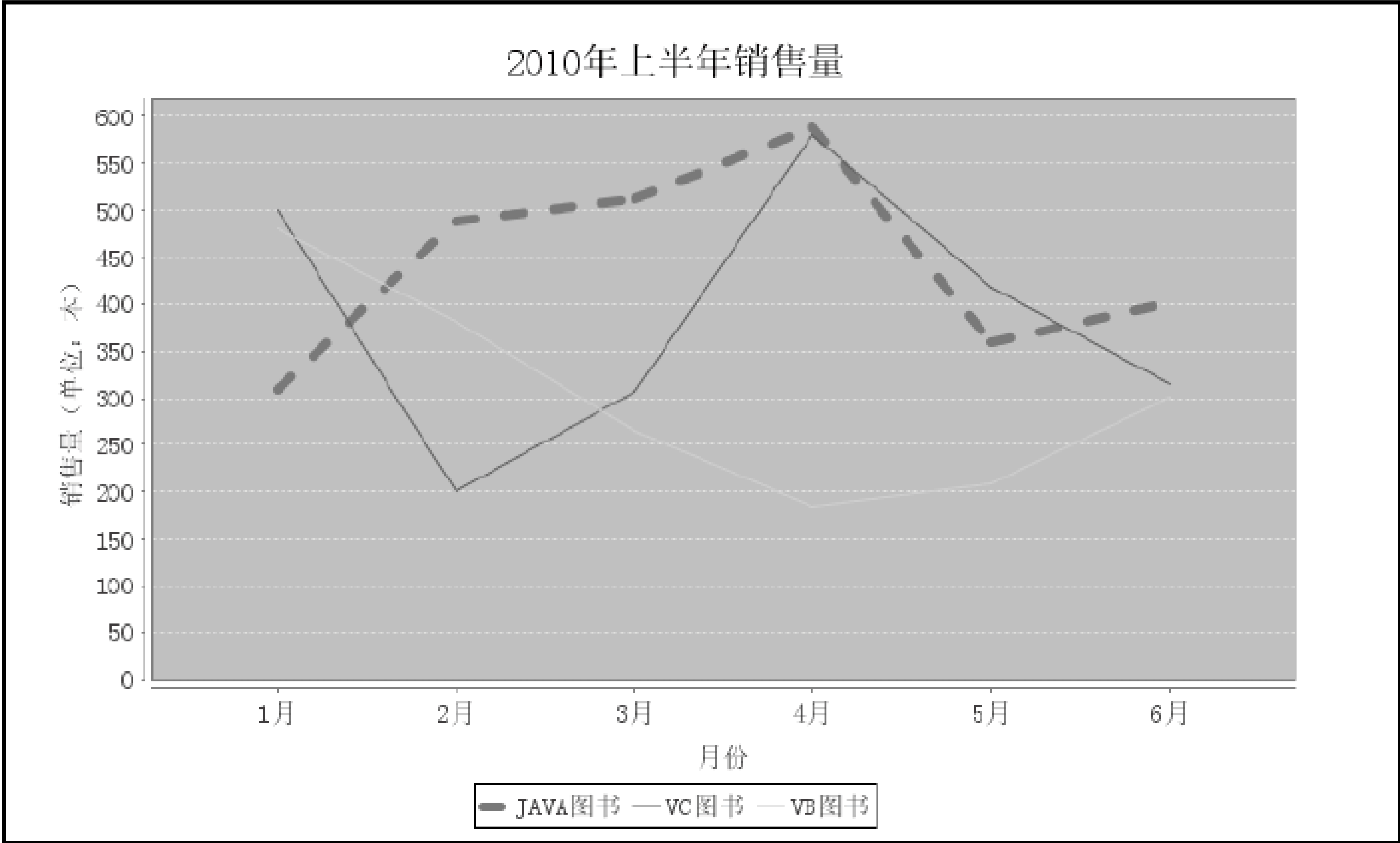


图 9.17　虚线折线图

关键技术

BasicStroke 类有多个构造方法，利用如下构造方法可以为折线图设置更丰富的图线笔触。

```
BasicStroke(float width, int cap, int join, float miterlimit, float dash[], float dash_phase)
```

参数说明

❶ width：表示笔触宽度。

❷ cap：表示笔触线条的端点样式。

❸ join：表示应用在路径线段相交处的样式。

❹ miterlimit：表示斜接处的剪裁限制，其值必须大于或等于 1.0f。

❺ dash：表示虚线模式的数组。

❻ dash_phase：表示开始虚线模式的偏移量。

（1）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，创建分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //系列关键字
```

```
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //分类关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createLineChart("2010 年上半年销售量",     //图表标题
                "月份",                                                           //X 轴标签
                "销售量（单位：本）",                                              //Y 轴标签
                dataset,                                                          //数据集
                PlotOrientation.VERTICAL,                                         //图表方向：水平、垂直
                true,                                                             //是否显示图例
                false,                                                            //是否生成工具栏提示
                false                                                             //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 LineAndShapeRenderer 实例，通过该实例设置折线为虚线。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        LineAndShapeRenderer renderer = (LineAndShapeRenderer) categoryPlot.getRenderer();
        //虚线
        renderer.setSeriesStroke(0, new BasicStroke(5, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND, 1, new float[] { 10.0f, 16.0f }, 0.0f));
}
```

■

心法领悟 249：BasicStroke 类的构造函数。

BasicStroke 类有多个构造函数，如本实例中的构造函数主要用来绘制虚线；如果只需要修改折线图的一些其他样式，如加粗笔触、修改端点样式，使用下面的构造函数即可。

```
BasicStroke(float width, int cap, int join)
```

参数说明

❶ width：表示笔触宽度。

❷ cap：表示笔触线条的端点样式。

❸ join：表示应用在路径线段相交处的样式。

实例 250　设置折线颜色

光盘位置：光盘\MR\09\250

高级

实用指数：★★★★★

实例说明

折线图的颜色丰富，用户可以根据实际情况修改折线颜色。本实例将演示如何修改折线图的默认颜色，运行结果如图 9.18 所示。

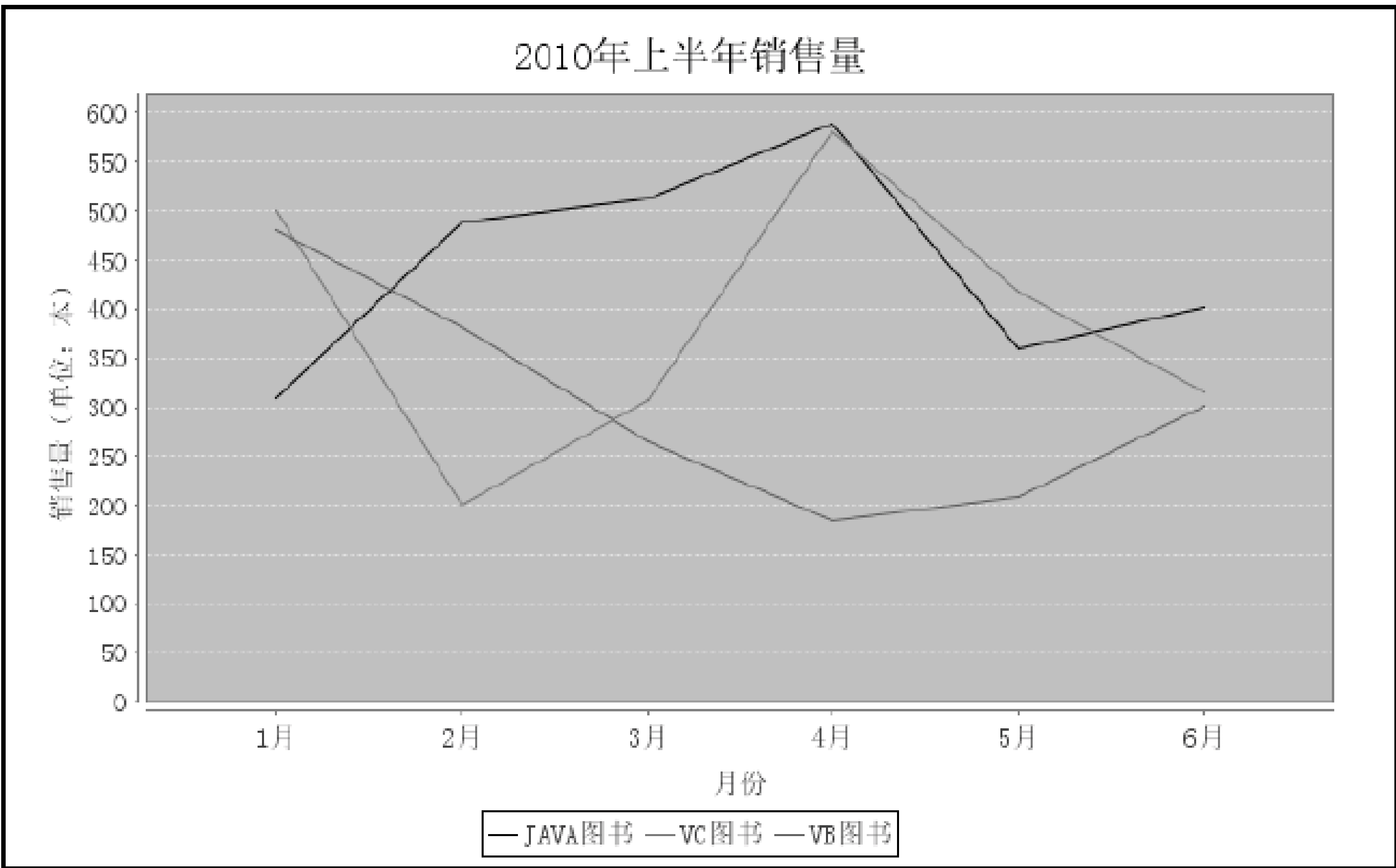


图 9.18　设置折线颜色

关键技术

使用 LineAndShapeRenderer 类的 setSeriesPaint()方法可以修改折线图的颜色。语法如下：

```
setSeriesPaint(int series, Paint paint)
```

参数说明

❶ series：表示折线图当前的系列索引。

❷ paint：表示准备设置系列的颜色。

（1）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，创建分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //系列关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //分类关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
```

```
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createLineChart("2010 年上半年销售量",        //图表标题
                "月份",                                                       //X 轴标签
                "销售量（单位：本）",                                          //Y 轴标签
                dataset,                                                      //数据集
                PlotOrientation.VERTICAL,                                     //图表方向：水平、垂直
                true,                                                         //是否显示图例
                false,                                                        //是否生成工具栏提示
                false                                                         //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取 LineAndShapeRenderer 实例，通过该实例设置第一个索引的折线为黑色。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        LineAndShapeRenderer renderer = (LineAndShapeRenderer) chart.getCategoryPlot().getRenderer();
        //设置显示颜色
        renderer.setSeriesPaint(0, Color.black);
}
```

秘笈心法

心法领悟 250：Color 类。

Color 类主要用于颜色设置。在该类中定义了一些常用的颜色常量，用户可以将其用于折线图中。例如，Color.black 表示黑色，Color.blue 表示蓝色，Color.cyan 表示青色，Color.gray 表示灰色等。

实例 251	3D 折线图 光盘位置：光盘\MR\09\251	高级 实用指数：★★★★★

实例说明

JFreeChart 不仅可以绘制普通折线图，还可以绘制 3D 折线图，使图表更具立体感。本实例将演示如何绘制

3D 折线图，运行结果如图 9.19 所示。

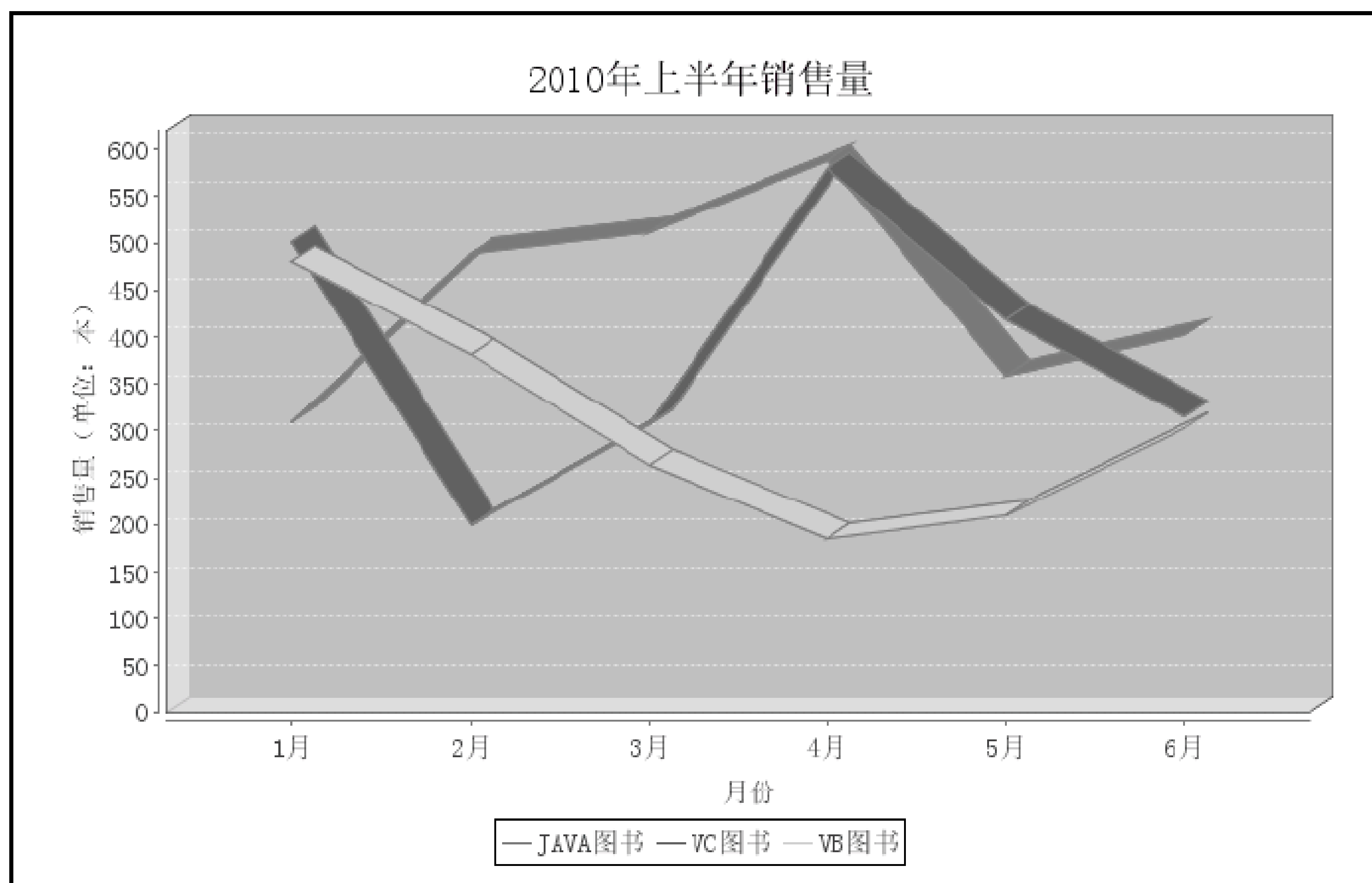


图 9.19　3D 折线图

关键技术

使用 ChartFactory 类的 createLineChart3D()方法可以创建 3D 折线图，创建完成后将返回一个 JFreeChart 对象。语法如下：

```
createLineChart3D (String title, String categoryAxisLabel, String valueAxisLabel, CategoryDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：图表的标题。

❷ categoryAxisLabel：图表类别标签，即 X 轴的名称。

❸ valueAxisLabel：图表数据标签，即 Y 轴的名称。

❹ dataset：折线图的数据集合。

❺ orientation：图表的显示方向。

❻ legend：表示是否使用图示。

❼ tooltips：表示是否生成工具栏提示。

❽ urls：表示是否生成 URL 链接。

（1）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，创建分类数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //系列关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //分类关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
```

```
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取数据集合，然后使用 ChartFactory 类的 createLineChart3D()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory. createLineChart3D ("2010 年上半年销售量",     //图表标题
                "月份",                                                              //X 轴标签
                "销售量（单位：本）",                                                  //Y 轴标签
                dataset,                                                             //数据集
                PlotOrientation.VERTICAL,                                            //图表方向：水平、垂直
                true,                                                                //是否显示图例
                false,                                                               //是否生成工具栏提示
                false                                                                //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改标题、X 轴、Y 轴、X 轴标签及 Y 轴标签等字体。代码如下：

```
private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

心法领悟 251：设置 3D 折线图墙的颜色。

在 3D 折线图中，可以使用 LineRenderer3D 类的 setWallPaint()方法设置 3D 折线图墙的颜色。语法如下：

```
setWallPaint(Paint paint)
```

参数说明

paint：表示 3D 折线图墙的颜色。



实例 252　XY 折线图

光盘位置：光盘\MR\09\252

高级

实用指数：★★★★★

实例说明

JFreeChart 可以绘制多种折线图，如分类折线图、XY 折线图。本实例将演示如何绘制 XY 折线图，运行结果如图 9.20 所示。

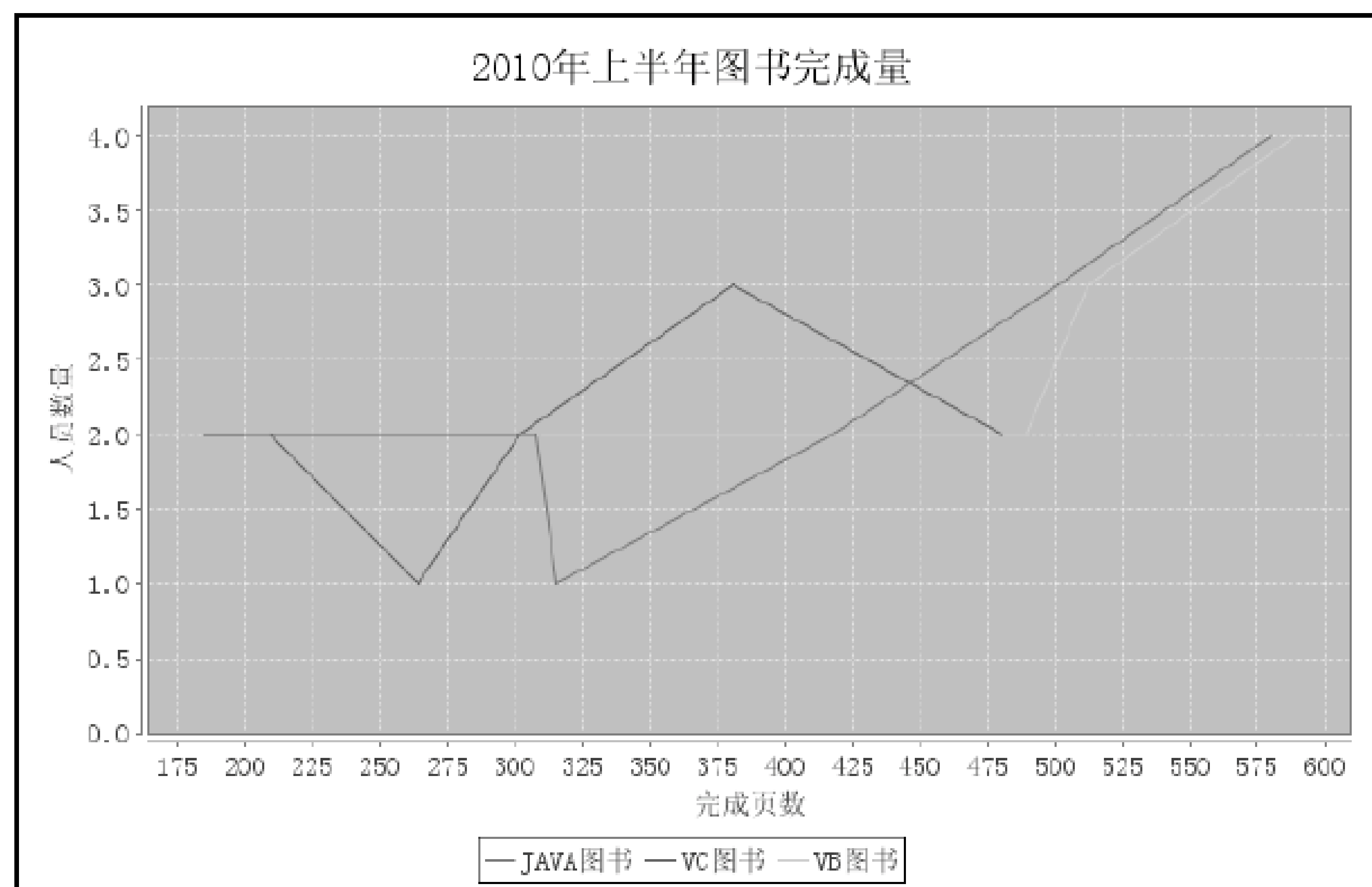


图 9.20　XY 折线图

关键技术

使用 ChartFactory 类的 createXYLineChart()方法可以创建基本的 XY 折线图，创建完成后将返回一个 JFreeChart 对象。语法如下：

```
createXYLineChart(String title, String xAxisLabel, String yAxisLabel, XYDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：图表的标题。
❷ xAxisLabel：图表 X 轴的标签内容。
❸ yAxisLabel：图表 Y 轴的标签内容。
❹ dataset：XY 折线图的数据集合。
❺ orientation：图表的显示方向。
❻ legend：表示是否使用图示。
❼ tooltips：表示是否生成工具栏提示。
❽ urls：表示是否生成 URL 链接。

（1）创建 getDataset()方法，通过 XYSeries 类向 XYSeriesCollection 类的实例中添加数据内容。代码如下：

```
private static IntervalXYDataset getDataset() {
        final XYSeries series1 = new XYSeries("JAVA 图书");
        final XYSeries series2 = new XYSeries("VC 图书");
        final XYSeries series3 = new XYSeries("VB 图书");
        series1.add(501, 3);
        series1.add(200, 2);
        series1.add(308, 2);
        series1.add(580, 4);
        series1.add(418, 2);
        series1.add(315, 1);
```

```
        series2.add(480, 2);
        series2.add(381, 3);
        series2.add(264, 1);
        series2.add(185, 2);
        series2.add(209, 2);
        series2.add(302, 2);
        series3.add(310, 2);
        series3.add(489, 2);
        series3.add(512, 3);
        series3.add(589, 4);
        series3.add(359, 2);
        series3.add(402, 2);
        final XYSeriesCollection dataset = new XYSeriesCollection();
        dataset.addSeries(series1);
        dataset.addSeries(series2);
        dataset.addSeries(series3);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取 IntervalXYDataset 数据集合，然后使用 ChartFactory 类的 createXYLineChart()方法根据数据集合生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        IntervalXYDataset dataset = getDataset();
        JFreeChart chart = ChartFactory.createXYLineChart("2010 年上半年图书完成量",      //图表标题
        "完成页数",                                                                      //X 轴标签
        "人员数量",                                                                      //Y 轴标签
        dataset,                                                                         //数据集
        PlotOrientation.VERTICAL,                                                        //图表方向：水平、垂直
        true,                                                                            //是否显示图例
        false,                                                                           //是否生成工具栏提示
        false                                                                            //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改标题、X 轴、Y 轴、X 轴标签及 Y 轴标签等字体。代码如下：

```
private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        XYPlot plot = chart.getXYPlot();
        NumberAxis domainAxis = (NumberAxis) plot.getDomainAxis();
        //X 轴字体
        domainAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        domainAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = plot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

心法领悟 252：添加 XY 折线图数据。

通常使用 XYSeries 类的 add()方法向 XYSeries 实例中添加 XY 折线图数据，再把 XYSeries 实例添加至数据集中。语法如下：

```
add(double x, double y)
```

参数说明

❶ x：表示 X 轴对应的数据值。

❷ y：表示 Y 轴对应的数据值。



实例 253　排序折线图

光盘位置：光盘\MR\09\253

高级

实用指数：★★★★★

实例说明

在绘制折线图时，首先要准备数据集用以填充。那么如何对数据集进行排序显示呢？本实例将演示如何将 X 轴进行升序排列，运行结果如图 9.21 所示。

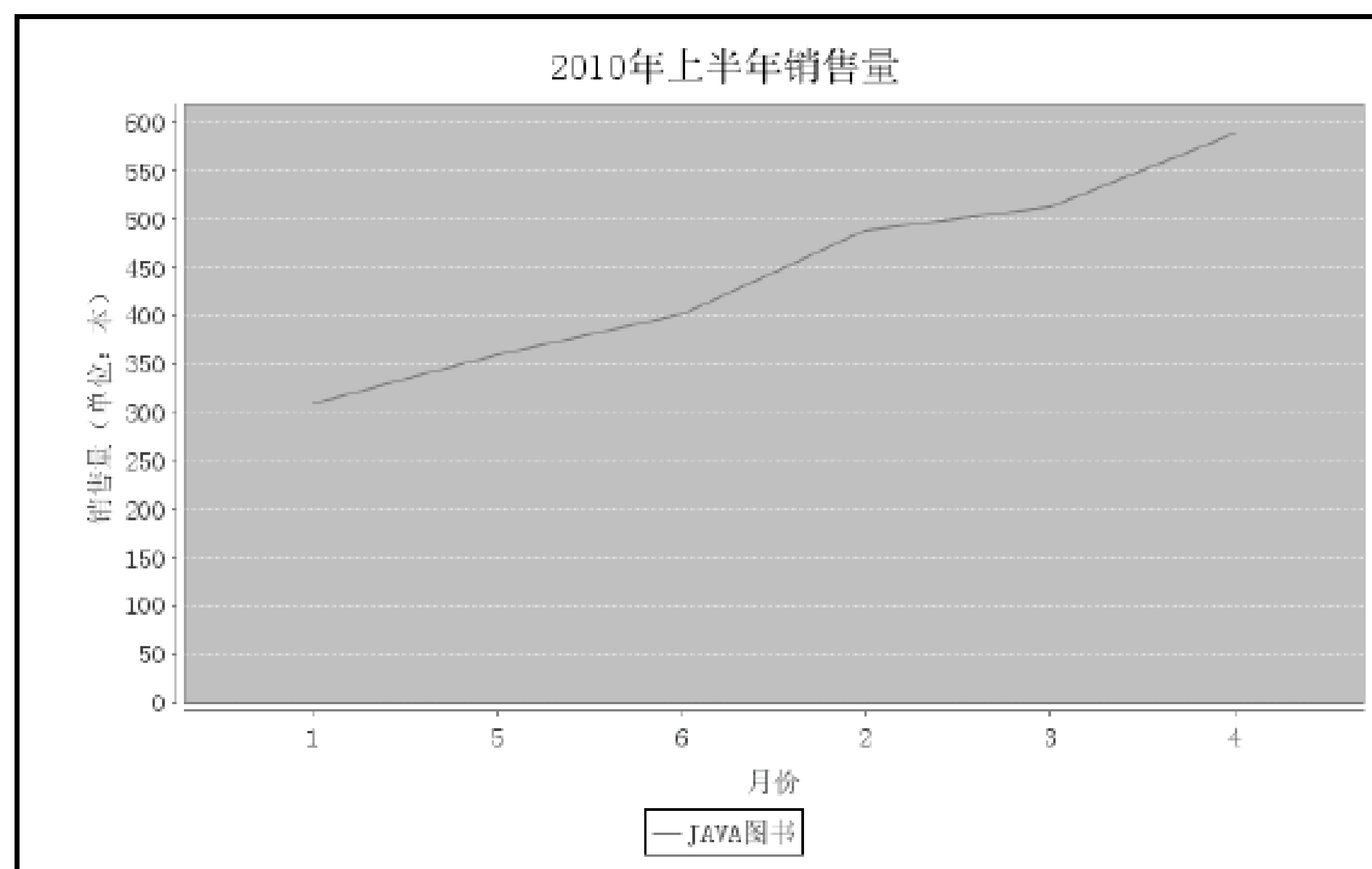


图 9.21　排序折线图

关键技术

使用 DefaultKeyedValues 类的 sortByValues()方法可以对数据集进行排序。语法如下：

```
sortByValues(SortOrder order)
```

参数说明

order：表示 DefaultKeyedValues 数据内容的排序方式。

设计过程

（1）创建 getCategoryDataset()方法，使用 DefaultKeyedValues 类创建分类数据集。代码如下：

```
private CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        keyedValues.addValue("1", 310);
        keyedValues.addValue("2", 489);
        keyedValues.addValue("3", 512);
        keyedValues.addValue("4", 589);
        keyedValues.addValue("5", 359);
        keyedValues.addValue("6", 402);
        //排序
        keyedValues.sortByValues(SortOrder.ASCENDING);
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("JAVA 图书", keyedValues);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中获取 CategoryDataset 数据集合，然后使用 ChartFactory 类的 createLineChart()方法生成 JFreeChart 对象。代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createLineChart("2010 年上半年销售量",        //图表标题
                "月份",                                                     //X 轴标签
                "销售量（单位：本）",                                         //Y 轴标签
                dataset,                                                    //数据集
                PlotOrientation.VERTICAL,                                   //图表方向：水平、垂直
                true,                                                       //是否显示图例
                false,                                                      //是否生成工具栏提示
```

```
            false                                                    //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改标题、X 轴、Y 轴、X 轴标签及 Y 轴标签等字体。代码如下：

```
private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        XYPlot plot = chart.getXYPlot();
        NumberAxis domainAxis = (NumberAxis) plot.getDomainAxis();
        //X 轴字体
        domainAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        domainAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = plot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

秘笈心法

心法领悟 253：对数据集进行排序。

利用 DefaultKeyedValues 类的 sortByValues()方法可以对数据集进行排序。该方法的参数为 SortOrder 类，此类中定义了两个常量——SortOrder.ASCENDING 和 SortOrder.DESCENDING，前者表示数据集升序排列，后者表示数据集降序排列。

9.3 时　序　图

实例 254	基本时序图 光盘位置：光盘\MR\09\254	高级 实用指数：★★★★★

实例说明

一般情况下，时序图使用时间轴表示某个分类的进展状况。本实例将演示如何绘制基本时序图，运行结果如图 9.22 所示。

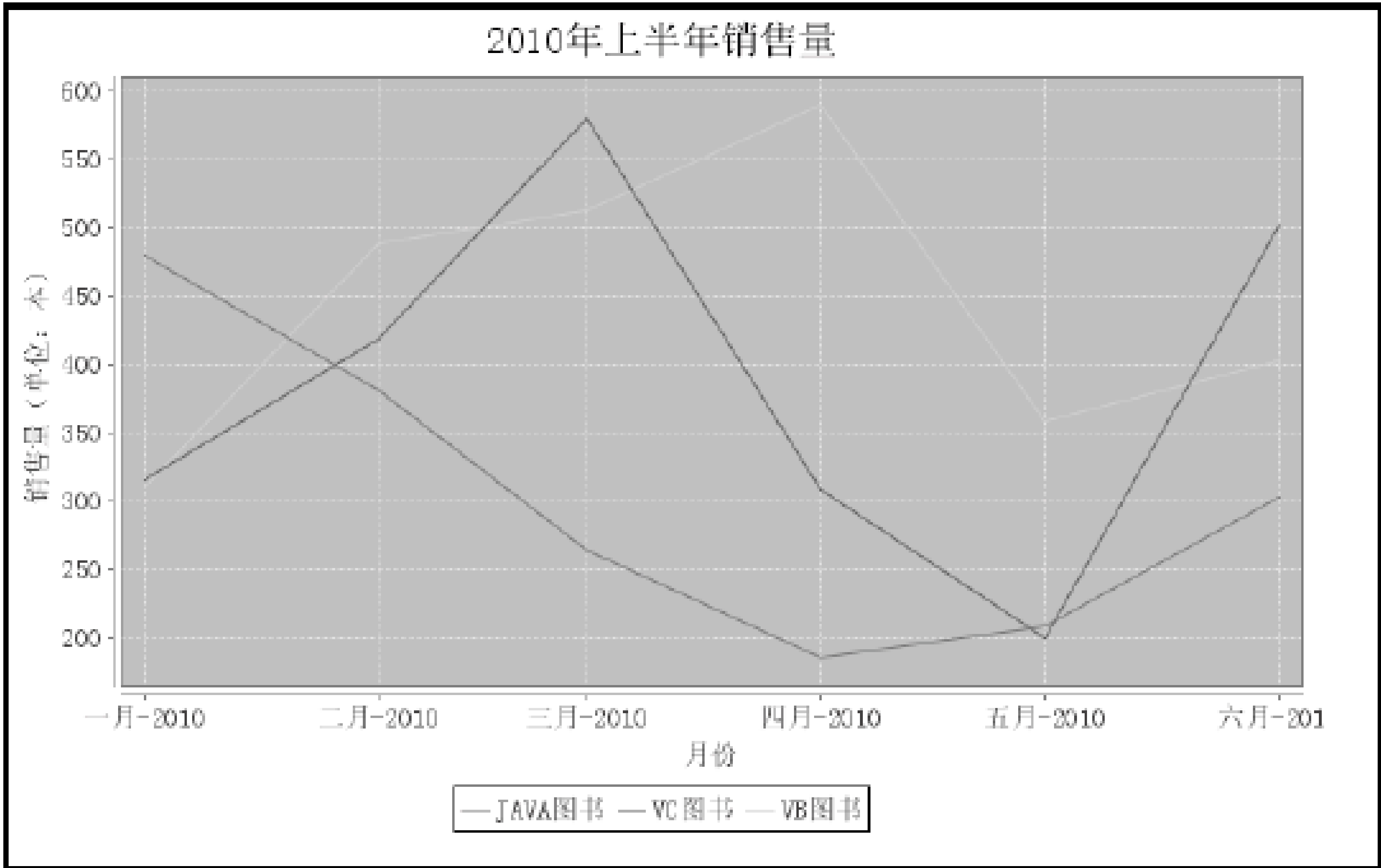


图 9.22　基本时序图

关键技术

使用 ChartFactory 类的 createTimeSeriesChart()方法可以创建一个基本时序图，创建完成后将返回一个 JFreeChart 对象。语法如下：

```
createTimeSeriesChart(String title, String timeAxisLabel, String valueAxisLabel, XYDataset dataset, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：图表的标题。

❷ timeAxisLabel：表示时间标签，一般用于 X 轴的名称。

❸ valueAxisLabel：表示范围标签，一般用于 Y 轴的名称。

❹ dataset：表示时序图的数据集合。

❺ legend：表示是否使用图示。

❻ tooltips：表示是否生成工具栏提示。

❼ urls：表示是否生成 URL 链接。

（1）创建 getDataset()方法，在该方法中使用 TimeSeriesCollection 类创建一个数据集。代码如下：

```
private static XYDataset getDataset() {
        final TimeSeries s1 = new TimeSeries("JAVA 图书");
        s1.add(new Month(1, 2010), 480);
        s1.add(new Month(2, 2010), 381);
        s1.add(new Month(3, 2010), 264);
        s1.add(new Month(4, 2010), 185);
        s1.add(new Month(5, 2010), 209);
        s1.add(new Month(6, 2010), 302);
        final TimeSeries s2 = new TimeSeries("VC 图书");
        s2.add(new Month(1, 2010), 315);
        s2.add(new Month(2, 2010), 418);
        s2.add(new Month(3, 2010), 580);
        s2.add(new Month(4, 2010), 308);
        s2.add(new Month(5, 2010), 200);
        s2.add(new Month(6, 2010), 501);
        final TimeSeries s3 = new TimeSeries("VB 图书");
        s3.add(new Month(1, 2010), 310);
        s3.add(new Month(2, 2010), 489);
        s3.add(new Month(3, 2010), 512);
        s3.add(new Month(4, 2010), 589);
        s3.add(new Month(5, 2010), 359);
        s3.add(new Month(6, 2010), 402);
        final TimeSeriesCollection dataset = new TimeSeriesCollection();
        dataset.addSeries(s1);
        dataset.addSeries(s2);
        dataset.addSeries(s3);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中调用 getDataset()方法获取数据集合，然后使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        XYDataset dataset = getDataset();
        JFreeChart chart = ChartFactory.createTimeSeriesChart("2010 年上半年销售量",  //图表标题
                "月份",                                                    //X 轴标签
                "销售量（单位：本）",                                        //Y 轴标签
                dataset,                                                   //数据集
                true,                                                      //是否显示图例
                false,                                                     //是否生成工具栏提示
                false                                                      //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改图表、标题、图示及坐标轴等字体。代码如下：

```
public static void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        XYPlot xyPlot = chart.getXYPlot();
        ValueAxis domainyAxis = xyPlot.getDomainAxis();
        //X 轴字体
        domainyAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        domainyAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis rangeAxis = xyPlot.getRangeAxis();
        //Y 轴字体
        rangeAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        rangeAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

秘笈心法

心法领悟 254：向数据集添加时序数据。

使用 TimeSeriesCollection 类的 addSeries()方法可以向数据集添加时序数据。语法如下：

```
addSeries(TimeSeries series)
```

参数说明

series：表示数据集中的时序数据。

实例 255 设置时间显示格式

光盘位置：光盘\MR\09\255

高级

实用指数：★★★★

实例说明

时序图中时间刻度的样式可以根据需求进行调整。本实例将演示如何格式化时序图中的时间样式，运行结果如图 9.23 所示。

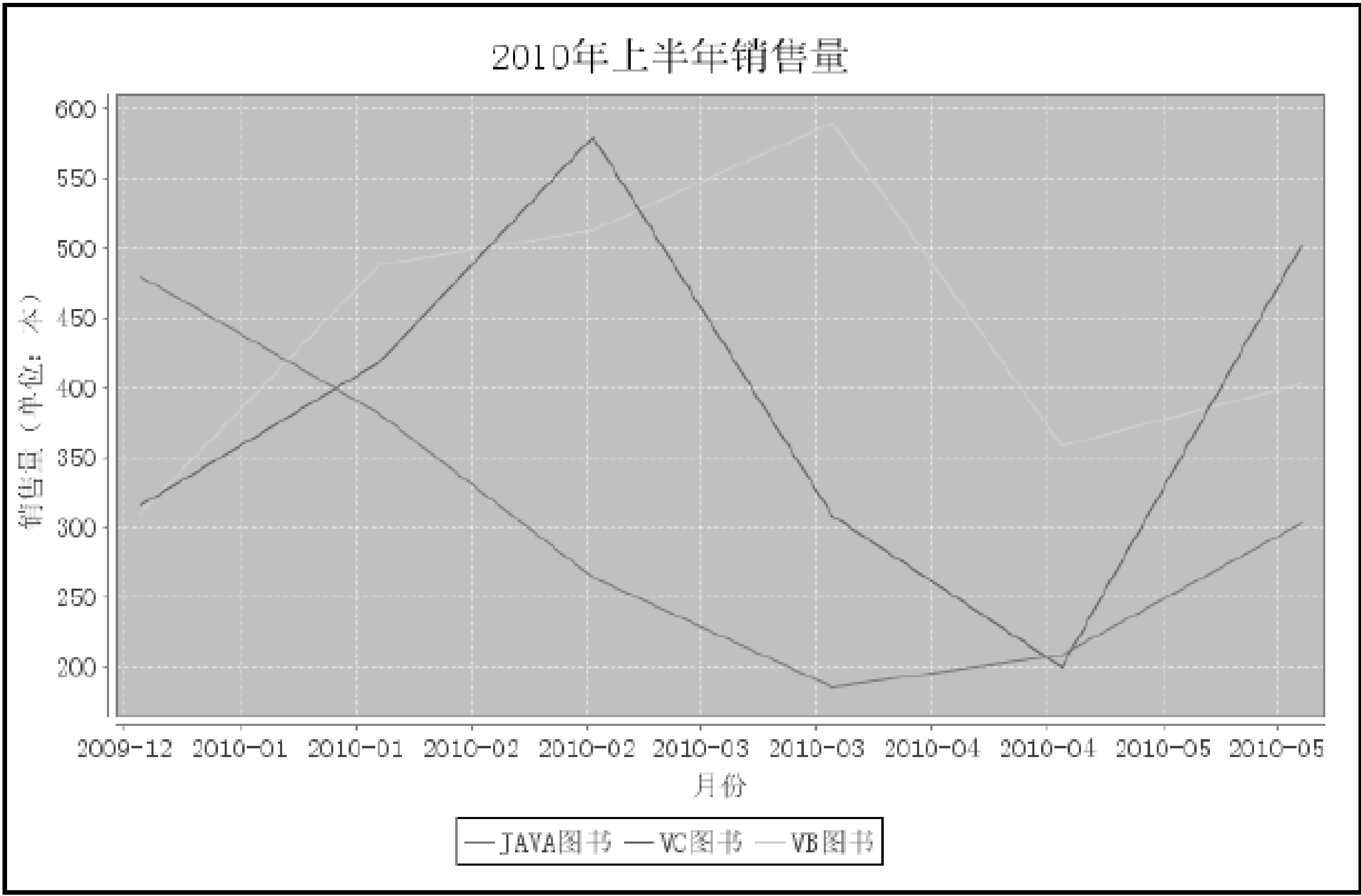


图 9.23　设置时间格式

使用 DateAxis 类的 setDateFormatOverride()方法可以格式化时间轴显示格式。语法如下：

```
setDateFormatOverride(DateFormat formatter)
```

参数说明

formatter：表示时序图的显示格式。

设计过程

（1）创建 getDataset()方法，在该方法中使用 TimeSeriesCollection 类创建一个数据集。代码如下：

```
private static XYDataset getDataset() {
	final TimeSeries s1 = new TimeSeries("JAVA 图书");
	s1.add(new Month(1, 2010), 480);
	s1.add(new Month(2, 2010), 381);
	s1.add(new Month(3, 2010), 264);
	s1.add(new Month(4, 2010), 185);
	s1.add(new Month(5, 2010), 209);
	s1.add(new Month(6, 2010), 302);
	final TimeSeries s2 = new TimeSeries("VC 图书");
	s2.add(new Month(1, 2010), 315);
	s2.add(new Month(2, 2010), 418);
	s2.add(new Month(3, 2010), 580);
	s2.add(new Month(4, 2010), 308);
	s2.add(new Month(5, 2010), 200);
	s2.add(new Month(6, 2010), 501);
	final TimeSeries s3 = new TimeSeries("VB 图书");
	s3.add(new Month(1, 2010), 310);
	s3.add(new Month(2, 2010), 489);
	s3.add(new Month(3, 2010), 512);
	s3.add(new Month(4, 2010), 589);
	s3.add(new Month(5, 2010), 359);
	s3.add(new Month(6, 2010), 402);
	final TimeSeriesCollection dataset = new TimeSeriesCollection();
	dataset.addSeries(s1);
	dataset.addSeries(s2);
	dataset.addSeries(s3);
	return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中调用 getDataset()方法获取数据集合，然后使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
	XYDataset dataset = getDataset();
	JFreeChart chart = ChartFactory.createTimeSeriesChart("2010 年上半年销售量",	//图表标题
		"月份",								//X 轴标签
		"销售量（单位：本）",						//Y 轴标签
		dataset,							//数据集
		true,								//是否显示图例
		false,								//是否生成工具栏提示
		false								//是否生成 URL 链接
	);
	return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取一个 XYPlot 实例，然后重新格式化时间格式。代码如下：

```
public static void updatePlot(JFreeChart chart) {
	//分类图表
	XYPlot xyPlot = chart.getXYPlot();
	DateAxis dateAxis = (DateAxis) xyPlot.getDomainAxis();
	//设置显示格式
	dateAxis.setDateFormatOverride(new SimpleDateFormat("yyyy-MM"));
}
```

心法领悟 255：重新设置时间格式。

使用 SimpleDateFormat 类的构造方法可以重新设置时间格式。语法如下：

```
SimpleDateFormat(String pattern)
```

参数说明

pattern：表示时间格式。

实例 256 添加双时间轴

光盘位置：光盘\MR\09\256

高级

实用指数：★★★★

实例说明

JFreeChart 可以为时序图添加第 2 个时间轴，用来表示另一个时间顺序。本实例将演示如何添加双时间轴，运行结果如图 9.24 所示。

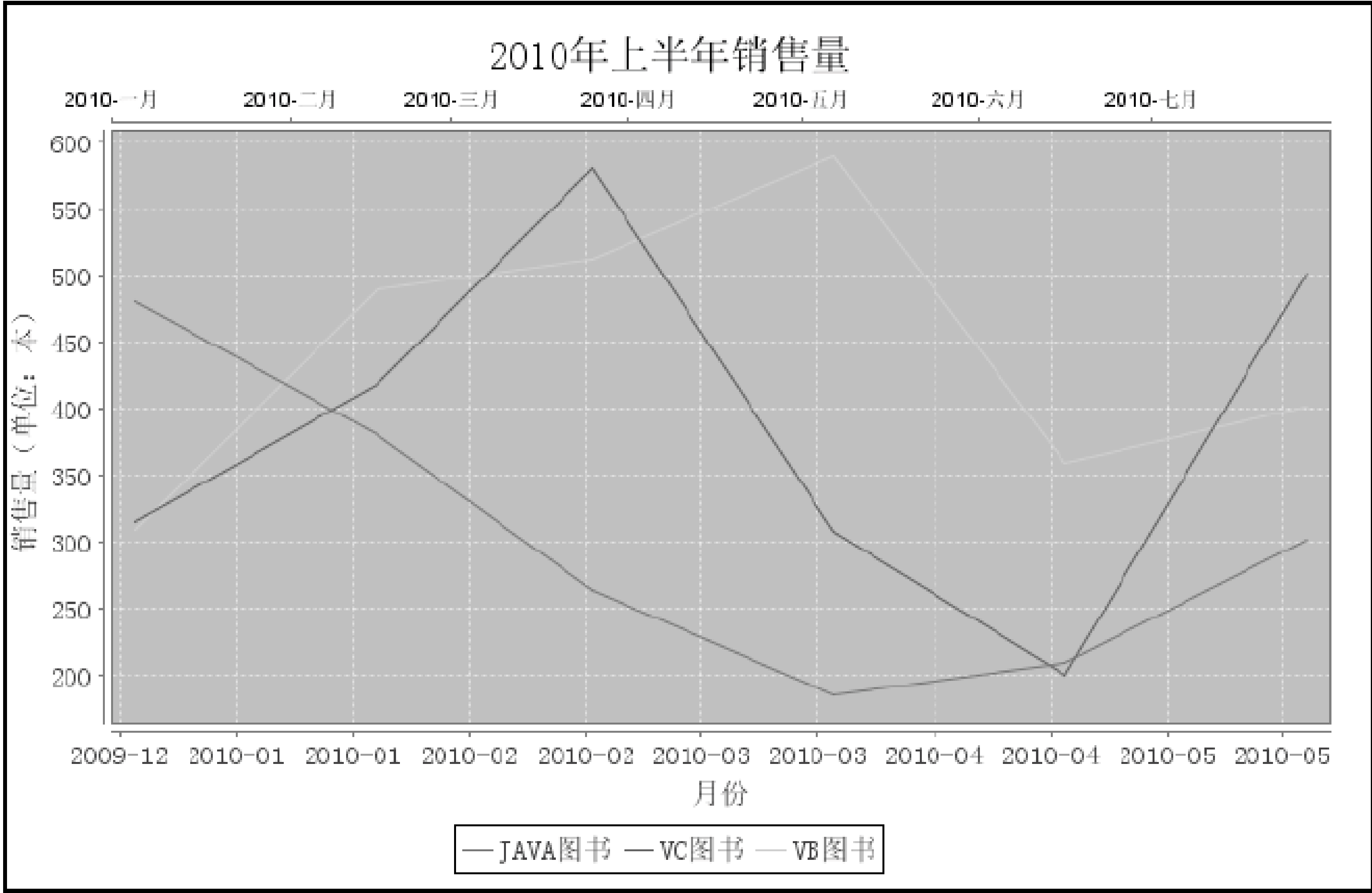


图 9.24　双时间轴

关键技术

使用 XYPlot 类的 setDomainAxis()方法可以为时序图新增一个时间轴。语法如下：

```
setDomainAxis(int index, ValueAxis axis)
```

参数说明

❶ index：表示时序图的索引。

❷ axis：表示要添加的指定索引的时间轴。

（1）创建 getDataset()方法，在该方法中使用 TimeSeriesCollection 类创建一个数据集。代码如下：

```
private static XYDataset getDataset() {
        final TimeSeries s1 = new TimeSeries("JAVA 图书");
        s1.add(new Month(1, 2010), 480);
        s1.add(new Month(2, 2010), 381);
        s1.add(new Month(3, 2010), 264);
        s1.add(new Month(4, 2010), 185);
        s1.add(new Month(5, 2010), 209);
        s1.add(new Month(6, 2010), 302);
        final TimeSeries s2 = new TimeSeries("VC 图书");
        s2.add(new Month(1, 2010), 315);
        s2.add(new Month(2, 2010), 418);
        s2.add(new Month(3, 2010), 580);
        s2.add(new Month(4, 2010), 308);
        s2.add(new Month(5, 2010), 200);
        s2.add(new Month(6, 2010), 501);
        final TimeSeries s3 = new TimeSeries("VB 图书");
        s3.add(new Month(1, 2010), 310);
```

```
        s3.add(new Month(2, 2010), 489);
        s3.add(new Month(3, 2010), 512);
        s3.add(new Month(4, 2010), 589);
        s3.add(new Month(5, 2010), 359);
        s3.add(new Month(6, 2010), 402);
        final TimeSeriesCollection dataset = new TimeSeriesCollection();
        dataset.addSeries(s1);
        dataset.addSeries(s2);
        dataset.addSeries(s3);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中调用 getDataset()方法获取数据集合，然后使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        XYDataset dataset = getDataset();
        JFreeChart chart = ChartFactory.createTimeSeriesChart("2010 年上半年销售量", //图表标题
        "月份",                                                    //X 轴标签
        "销售量（单位：本）",                                        //Y 轴标签
        dataset,                                                   //数据集
        true,                                                      //是否显示图例
        false,                                                     //是否生成工具栏提示
        false                                                      //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取一个 XYPlot 实例，为图表新增一个时间轴。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        XYPlot xyPlot = chart.getXYPlot();
        DateAxis dateAxis = (DateAxis) xyPlot.getDomainAxis();
        //设置显示格式
        dateAxis.setDateFormatOverride(new SimpleDateFormat("yyyy-MM"));
        //添加时间轴
        DateAxis dateAxis1 = new DateAxis();
        //添加时间范围
        dateAxis1.setRange(new Month(1, 2010).getStart(), new Month(7, 2010).getEnd());
        //设置时间表格
        dateAxis1.setDateFormatOverride(new SimpleDateFormat("yyyy-MMM"));
        xyPlot.setDomainAxis(1,dateAxis1);

}
```

秘笈心法

心法领悟 256：为时间轴添加时间范围。

使用 DateAxis 类的 setRange()方法可以为时间轴添加时间范围。语法如下：

```
setRange(Date lower, Date upper)
```

参数说明

❶ lower：表示时间轴的开始时间。

❷ upper：表示时间轴的结束时间。

实例 257	设置双时间轴位置 光盘位置：光盘\MR\09\257	高级 实用指数：★★★

实例说明

JFreeChart 可以新增时间轴，新增的时间轴位置可以根据需要进行设置。本实例将演示如何设置双时间轴位置，运行结果如图 9.25 所示。

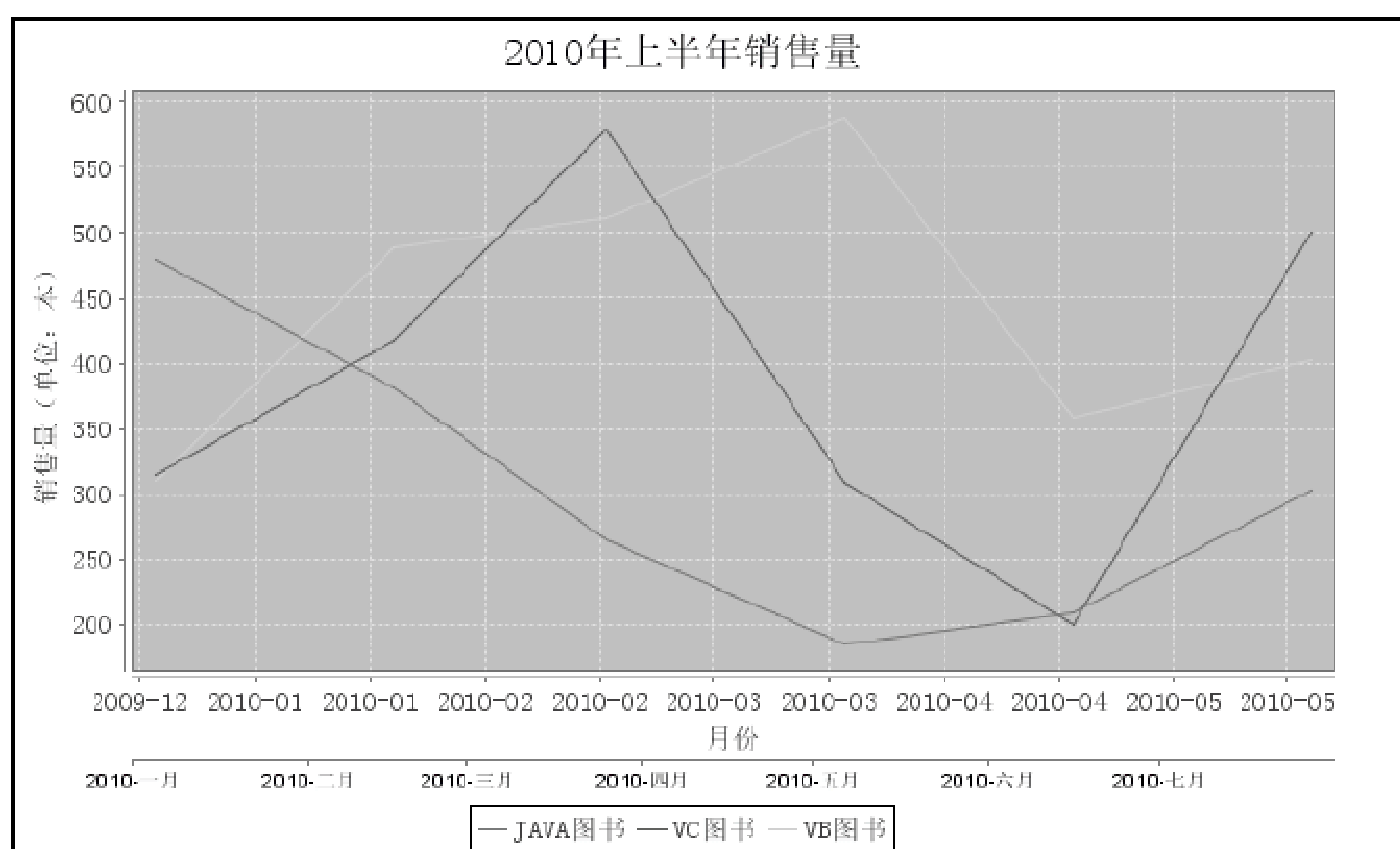


图 9.25　设置双时间轴位置

关键技术

使用 XYPlot 类的 setDomainAxisLocation()方法可以为新增的时间轴设置显示位置。语法如下：

```
setDomainAxisLocation(int index, AxisLocation location)
```

参数说明

❶ index：表示时间轴的索引。

❷ location：表示要设置时间轴的显示位置。

（1）创建 getDataset()方法，在该方法中使用 TimeSeriesCollection 类创建一个数据集。代码如下：

```
private static XYDataset getDataset() {
        final TimeSeries s1 = new TimeSeries("JAVA 图书");
        s1.add(new Month(1, 2010), 480);
        s1.add(new Month(2, 2010), 381);
        s1.add(new Month(3, 2010), 264);
        s1.add(new Month(4, 2010), 185);
        s1.add(new Month(5, 2010), 209);
        s1.add(new Month(6, 2010), 302);
        final TimeSeries s2 = new TimeSeries("VC 图书");
        s2.add(new Month(1, 2010), 315);
        s2.add(new Month(2, 2010), 418);
        s2.add(new Month(3, 2010), 580);
        s2.add(new Month(4, 2010), 308);
        s2.add(new Month(5, 2010), 200);
        s2.add(new Month(6, 2010), 501);
        final TimeSeries s3 = new TimeSeries("VB 图书");
        s3.add(new Month(1, 2010), 310);
        s3.add(new Month(2, 2010), 489);
        s3.add(new Month(3, 2010), 512);
        s3.add(new Month(4, 2010), 589);
        s3.add(new Month(5, 2010), 359);
        s3.add(new Month(6, 2010), 402);
        final TimeSeriesCollection dataset = new TimeSeriesCollection();
        dataset.addSeries(s1);
        dataset.addSeries(s2);
        dataset.addSeries(s3);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中调用 getDataset()方法获取数据集合，然后使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        XYDataset dataset = getDataset();
        JFreeChart chart = ChartFactory.createTimeSeriesChart("2010 年上半年销售量", //图表标题
```

```
            "月份",                   //X 轴标签
            "销售量（单位：本）",      //Y 轴标签
            dataset,                  //数据集
            true,                     //是否显示图例
            false,                    //是否生成工具栏提示
            false                     //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取一个 XYPlot 实例，为图表新增一个时间轴，并且设置时间轴在图表底部或者左侧。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        XYPlot xyPlot = chart.getXYPlot();
        DateAxis dateAxis = (DateAxis) xyPlot.getDomainAxis();
        //设置显示格式
        dateAxis.setDateFormatOverride(new SimpleDateFormat("yyyy-MM"));
        //添加时间轴
        DateAxis dateAxis1 = new DateAxis();
        dateAxis1.setRange(new Month(1, 2010).getStart(), new Month(7, 2010).getEnd());
        dateAxis1.setDateFormatOverride(new SimpleDateFormat("yyyy-MMM"));
        xyPlot.setDomainAxis(1,dateAxis1);
        //设置时间轴显示位置
        xyPlot.setDomainAxisLocation(1, AxisLocation.BOTTOM_OR_LEFT);
}
```

秘笈心法

心法领悟 257：时间轴位置常量。

在 AxisLocation 类中定义了时间轴的位置常量，AxisLocation.BOTTOM_OR_LEFT 表示时间轴位于图表底部或者左侧，AxisLocation.TOP_OR_RIGHT 表示时间轴位于图表顶部或者右侧。

实例 258　动态显示十字标记

光盘位置：光盘\MR\09\258

高级

实用指数：★★★★★

绘制完时序图后，单击 X 轴刻度与 Y 轴刻度相交处，图表上会显示出十字标记。本实例将演示如何在单击处显示 X、Y 轴交叉标记，运行结果如图 9.26 所示。

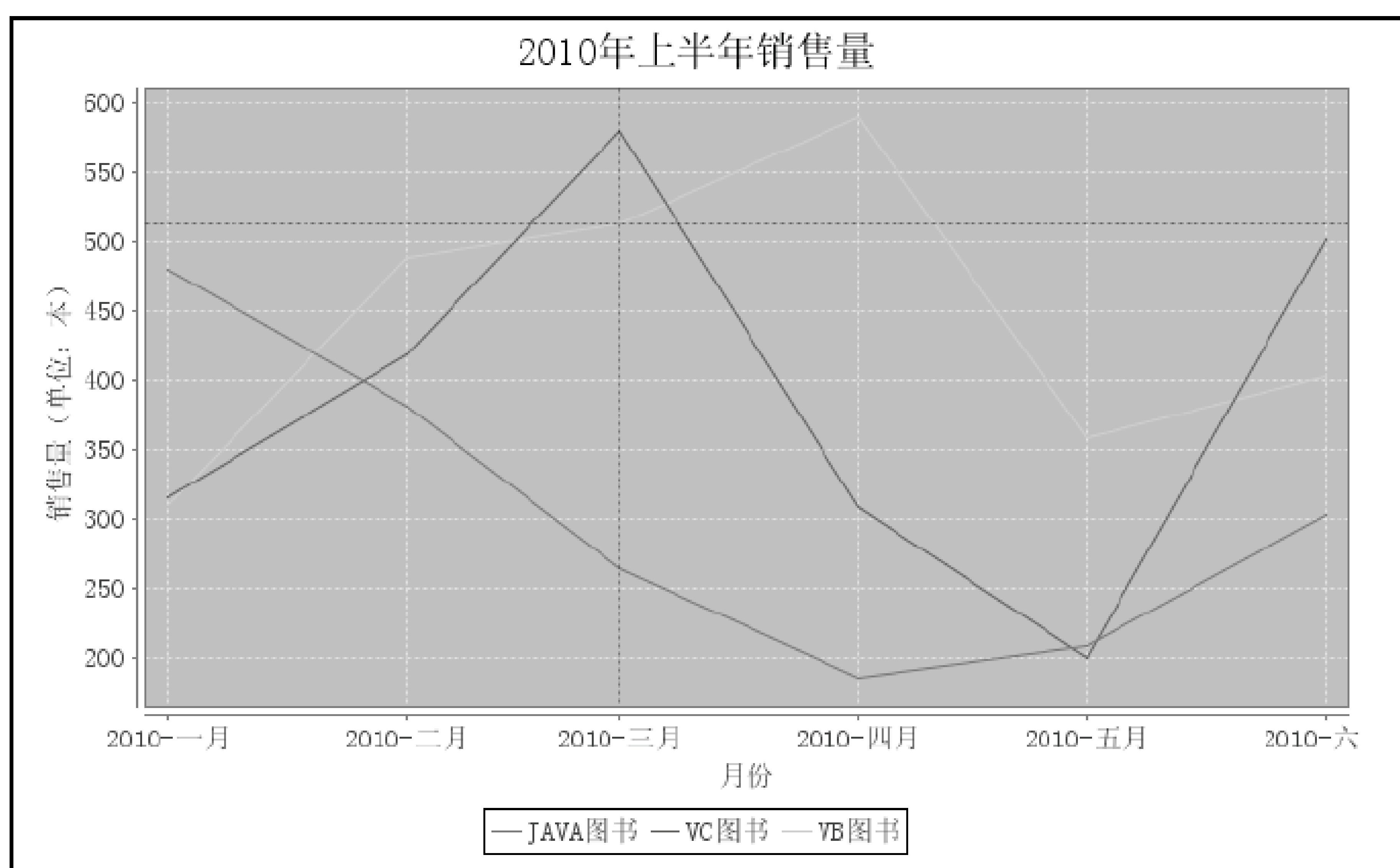


图 9.26　显示十字标记

关键技术

（1）使用 XYPlot 类的 setDomainCrosshairVisible()方法可以显示出时间轴上的标记。语法如下：

```
setDomainCrosshairVisible(boolean flag)
```

参数说明

flag：表示是否显示 X 轴标记。flag 为 true 时，表示显示 X 轴标记；flag 为 false 时，表示不显示 X 轴标记。

（2）使用 XYPlot 类的 setRangeCrosshairVisible()方法可以显示出范围轴上的标记。语法如下：

```
setRangeCrosshairVisible (boolean flag)
```

参数说明

flag：表示是否显示 Y 轴标记。flag 为 true 时，表示显示 Y 轴标记；flag 为 false 时，表示不显示 Y 轴标记。

（1）创建 getDataset()方法，在该方法中使用 TimeSeriesCollection 类创建一个数据集。代码如下：

```
private static XYDataset getDataset() {
        final TimeSeries s1 = new TimeSeries("JAVA 图书");
        s1.add(new Month(1, 2010), 480);
        s1.add(new Month(2, 2010), 381);
        s1.add(new Month(3, 2010), 264);
        s1.add(new Month(4, 2010), 185);
        s1.add(new Month(5, 2010), 209);
        s1.add(new Month(6, 2010), 302);
        final TimeSeries s2 = new TimeSeries("VC 图书");
        s2.add(new Month(1, 2010), 315);
        s2.add(new Month(2, 2010), 418);
        s2.add(new Month(3, 2010), 580);
        s2.add(new Month(4, 2010), 308);
        s2.add(new Month(5, 2010), 200);
        s2.add(new Month(6, 2010), 501);
        final TimeSeries s3 = new TimeSeries("VB 图书");
        s3.add(new Month(1, 2010), 310);
        s3.add(new Month(2, 2010), 489);
        s3.add(new Month(3, 2010), 512);
        s3.add(new Month(4, 2010), 589);
        s3.add(new Month(5, 2010), 359);
        s3.add(new Month(6, 2010), 402);
        final TimeSeriesCollection dataset = new TimeSeriesCollection();
        dataset.addSeries(s1);
        dataset.addSeries(s2);
        dataset.addSeries(s3);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中调用 getDataset()方法获取数据集合，然后使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        XYDataset dataset = getDataset();
        JFreeChart chart = ChartFactory.createTimeSeriesChart("2010 年上半年销售量", //图表标题
                "月份",                                              //X 轴标签
                "销售量（单位：本）",                                   //Y 轴标签
                dataset,                                             //数据集
                true,                                                //是否显示图例
                false,                                               //是否生成工具栏提示
                false                                                //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取一个 XYPlot 实例，设置图表根据单击情况显示十字标记。代码如下：

```
private void updatePlot(JFreeChart chart) {
        //分类图表
```

```
        XYPlot xyPlot = chart.getXYPlot();
        //显示十字标记
        xyPlot.setDomainCrosshairVisible(true);
        xyPlot.setRangeCrosshairVisible(true);
        DateAxis dateAxis = (DateAxis) xyPlot.getDomainAxis();
        //设置显示格式
        dateAxis.setDateFormatOverride(new SimpleDateFormat("yyyy-MMM"));
}
```

秘笈心法

心法领悟 258：设置 X、Y 轴的标记颜色。

（1）使用 XYPlot 类的 setDomainCrosshairPaint()方法可以设置 X 轴的标记颜色。语法如下：

```
setDomainCrosshairPaint (Paint paint)
```

参数说明

paint：表示 X 轴标记的颜色。

（2）使用 XYPlot 类的 setRangeCrosshairPaint()方法可以设置 Y 轴的标记颜色。语法如下：

```
setRangeCrosshairPaint(Paint paint)
```

参数说明

paint：表示 Y 轴标记的颜色。

实例 259	添加 Y 轴标记 光盘位置：光盘\MR\09\259	高级 实用指数：★★★★

实例说明

绘制时序图时，通过为 Y 轴添加标记可以定位图表数据。本实例将演示如何为 Y 轴添加标记，运行结果如图 9.27 所示。

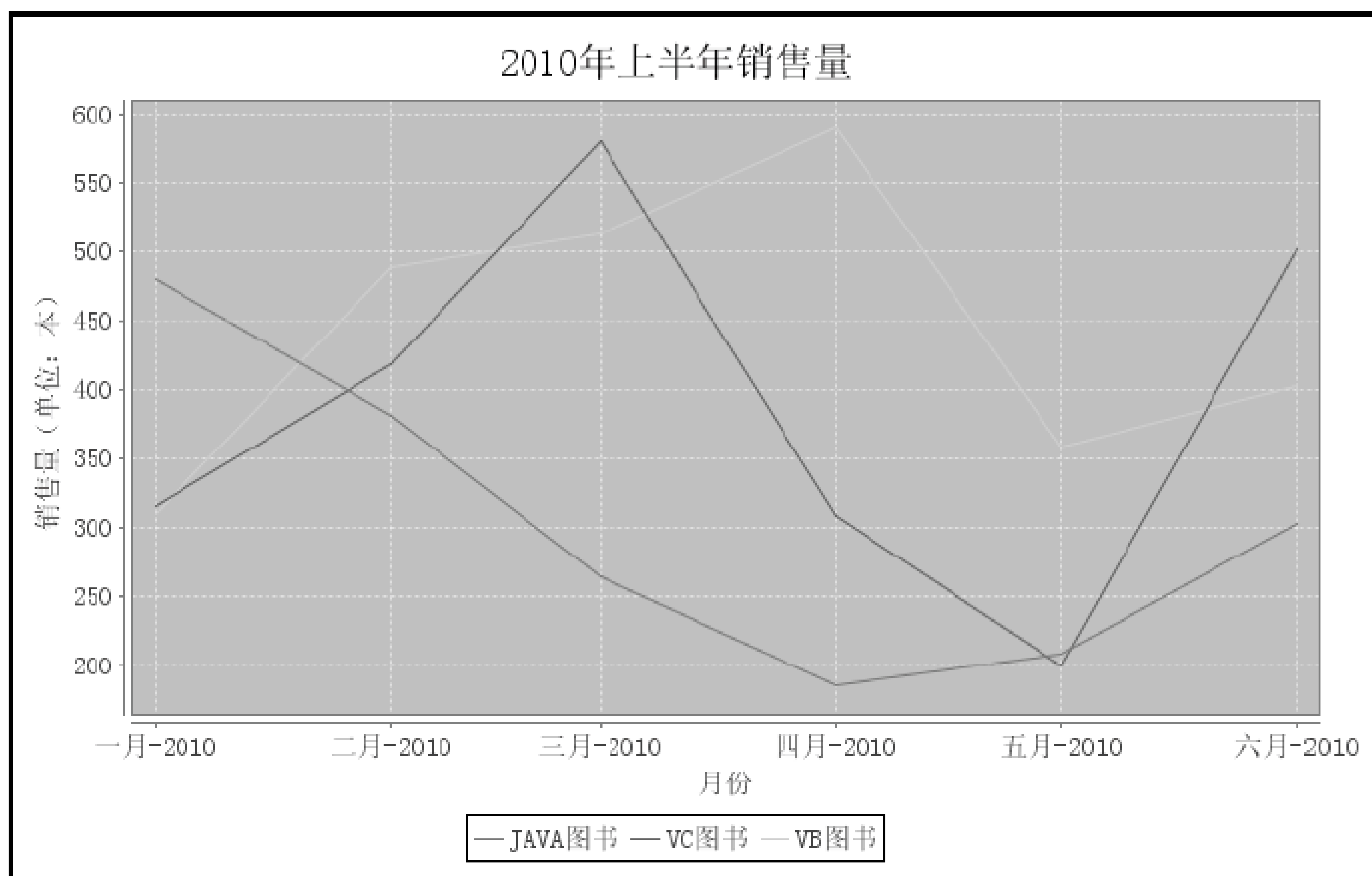


图 9.27　添加 Y 轴标记

使用 XYPlot 类的 addRangeMarker()方法可以绘制 Y 轴上的标记。语法如下：

```
addRangeMarker(Marker marker)
```

参数说明

marker：表示 Y 轴上的标记。

设计过程

（1）创建 getDataset()方法，在该方法中使用 TimeSeriesCollection 类创建一个数据集。代码如下：

```
private static XYDataset getDataset() {
        final TimeSeries s1 = new TimeSeries("JAVA 图书");
        s1.add(new Month(1, 2010), 480);
        s1.add(new Month(2, 2010), 381);
        s1.add(new Month(3, 2010), 264);
        s1.add(new Month(4, 2010), 185);
        s1.add(new Month(5, 2010), 209);
        s1.add(new Month(6, 2010), 302);
        final TimeSeries s2 = new TimeSeries("VC 图书");
        s2.add(new Month(1, 2010), 315);
        s2.add(new Month(2, 2010), 418);
        s2.add(new Month(3, 2010), 580);
        s2.add(new Month(4, 2010), 308);
        s2.add(new Month(5, 2010), 200);
        s2.add(new Month(6, 2010), 501);
        final TimeSeries s3 = new TimeSeries("VB 图书");
        s3.add(new Month(1, 2010), 310);
        s3.add(new Month(2, 2010), 489);
        s3.add(new Month(3, 2010), 512);
        s3.add(new Month(4, 2010), 589);
        s3.add(new Month(5, 2010), 359);
        s3.add(new Month(6, 2010), 402);
        final TimeSeriesCollection dataset = new TimeSeriesCollection();
        dataset.addSeries(s1);
        dataset.addSeries(s2);
        dataset.addSeries(s3);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中调用 getDataset()方法获取数据集合，然后使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        XYDataset dataset = getDataset();
        JFreeChart chart = ChartFactory.createTimeSeriesChart("2010 年上半年销售量",  //图表标题
        "月份",                                                                    //X 轴标签
        "销售量（单位：本）",                                                      //Y 轴标签
        dataset,                                                                   //数据集
        true,                                                                      //是否显示图例
        false,                                                                     //是否生成工具栏提示
        false                                                                      //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取一个 XYPlot 实例，并且在 Y 轴上添加标记。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        XYPlot xyPlot = chart.getXYPlot();
        //添加 Y 轴标记
        ValueMarker marker = new ValueMarker(450.0);
        marker.setPaint(Color.orange);
        xyPlot.addRangeMarker(marker);
}
```

秘笈心法

心法领悟 259：创建数据轴的标记。

使用 ValueMarker 的构造方法可以创建数据轴的标记实例。语法如下：

```
ValueMarker(double value)
```

参数说明

value：表示 Y 轴标记上的数据值。



实例 260　添加 X 轴标记

光盘位置：光盘\MR\09\260

高级

实用指数：★★★★

实例说明

绘制时序图时，通过为 X 轴添加标记可以定位时间轴。本实例将演示如何为 X 轴添加标记，运行结果如图 9.28 所示。

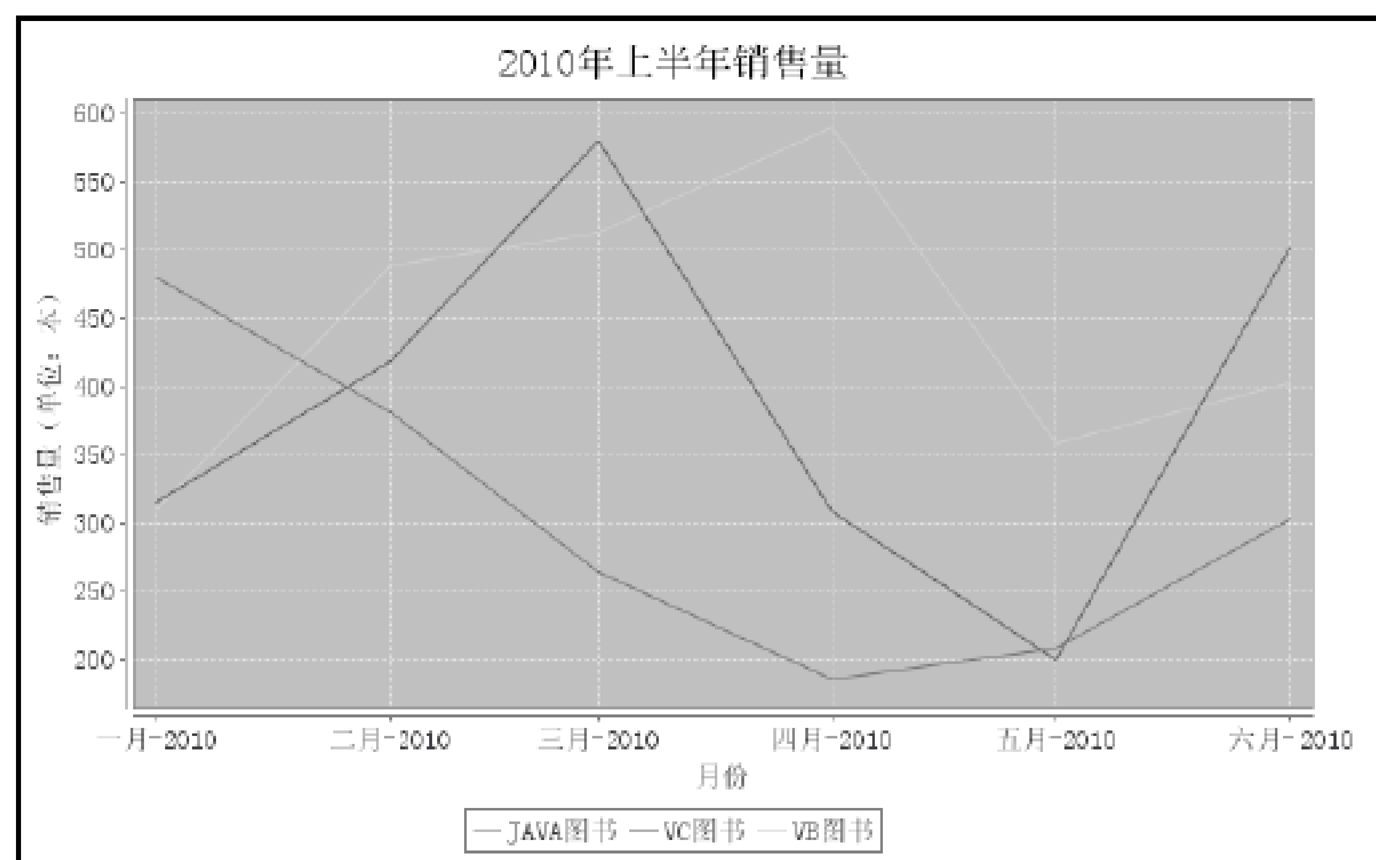


图 9.28　显示 X 轴标记

关键技术

使用 XYPlot 类的 addDomainMarker()方法可以绘制 X 轴上的标记。语法如下：

```
addDomainMarker(Marker marker)
```

参数说明

marker：表示 X 轴上的标记。

设计过程

（1）创建 getDataset()方法，在该方法中使用 TimeSeriesCollection 类创建一个数据集。代码如下：

```
private static XYDataset getDataset() {
        final TimeSeries s1 = new TimeSeries("JAVA 图书");
        s1.add(new Month(1, 2010), 480);
        s1.add(new Month(2, 2010), 381);
        s1.add(new Month(3, 2010), 264);
        s1.add(new Month(4, 2010), 185);
        s1.add(new Month(5, 2010), 209);
        s1.add(new Month(6, 2010), 302);
        final TimeSeries s2 = new TimeSeries("VC 图书");
        s2.add(new Month(1, 2010), 315);
        s2.add(new Month(2, 2010), 418);
        s2.add(new Month(3, 2010), 580);
        s2.add(new Month(4, 2010), 308);
        s2.add(new Month(5, 2010), 200);
        s2.add(new Month(6, 2010), 501);
        final TimeSeries s3 = new TimeSeries("VB 图书");
        s3.add(new Month(1, 2010), 310);
        s3.add(new Month(2, 2010), 489);
        s3.add(new Month(3, 2010), 512);
        s3.add(new Month(4, 2010), 589);
        s3.add(new Month(5, 2010), 359);
        s3.add(new Month(6, 2010), 402);
        final TimeSeriesCollection dataset = new TimeSeriesCollection();
        dataset.addSeries(s1);
        dataset.addSeries(s2);
        dataset.addSeries(s3);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中调用 getDataset()方法获取数据集合，然后使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        XYDataset dataset = getDataset();
        JFreeChart chart = ChartFactory.createTimeSeriesChart("2010 年上半年销售量",  //图表标题
        "月份",                                                                        //X 轴标签
        "销售量（单位：本）",                                                          //Y 轴标签
        dataset,                                                                       //数据集
        true,                                                                          //是否显示图例
        false,                                                                         //是否生成工具栏提示
        false                                                                          //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取一个 XYPlot 实例，并且在 X 轴上添加标记。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        XYPlot xyPlot = chart.getXYPlot();
        Quarter quarter = new Quarter(2, 2010);
        ValueMarker marker = new ValueMarker(quarter.getFirstMillisecond());
        marker.setPaint(Color.ORANGE);
        //添加标记
        xyPlot.addDomainMarker(marker);
}
```

秘笈心法

心法领悟 260：创建季度。

使用 Quarter 的构造方法可以创建季度实例。语法如下：

```
Quarter(int quarter, int year)
```

参数说明

❶ quarter：表示季度数值。

❷ year：表示年度数值。

实例 261 设置刻度单位

光盘位置：光盘\MR\09\261

高级

实用指数：★★★★

绘制时序图时，X 轴的刻度可以根据不同需要设置不同的刻度单位。本实例将演示如何设置 X 轴的刻度单位，运行结果如图 9.29 所示。

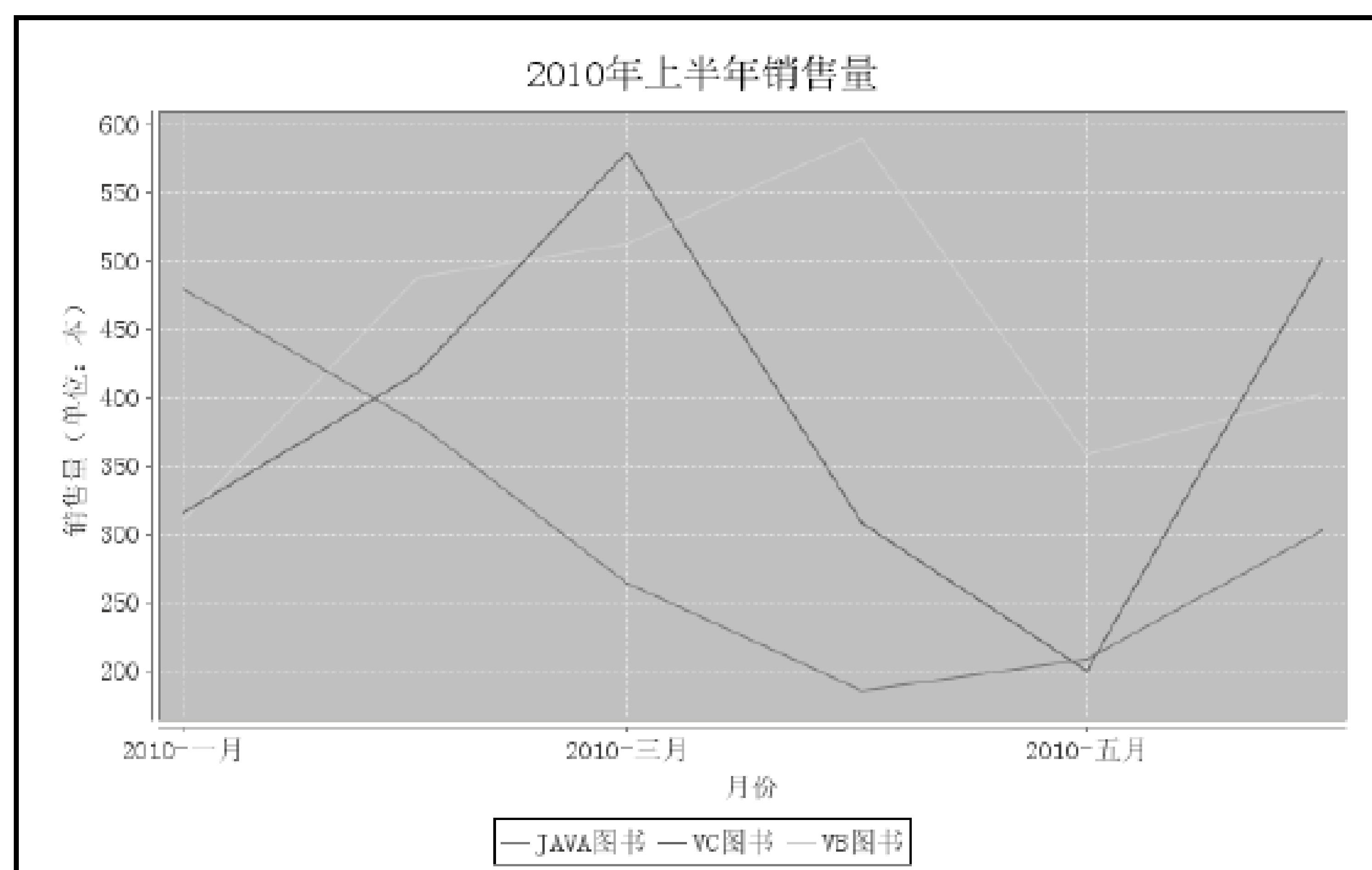


图 9.29　设置刻度单位

关键技术

使用 DateAxis 类的 setTickUnit()方法可以设置 X 轴上刻度的单位。语法如下：

```
setTickUnit(DateTickUnit unit)
```

参数说明

unit：日期类型，表示 X 轴刻度单位的实例。

（1）创建 getDataset()方法，在该方法中使用 TimeSeriesCollection 类创建一个数据集。代码如下：

```
private static XYDataset getDataset() {
        final TimeSeries s1 = new TimeSeries("JAVA 图书");
        s1.add(new Month(1, 2010), 480);
        s1.add(new Month(2, 2010), 381);
        s1.add(new Month(3, 2010), 264);
        s1.add(new Month(4, 2010), 185);
        s1.add(new Month(5, 2010), 209);
        s1.add(new Month(6, 2010), 302);
        final TimeSeries s2 = new TimeSeries("VC 图书");
        s2.add(new Month(1, 2010), 315);
        s2.add(new Month(2, 2010), 418);
        s2.add(new Month(3, 2010), 580);
        s2.add(new Month(4, 2010), 308);
        s2.add(new Month(5, 2010), 200);
        s2.add(new Month(6, 2010), 501);
        final TimeSeries s3 = new TimeSeries("VB 图书");
        s3.add(new Month(1, 2010), 310);
        s3.add(new Month(2, 2010), 489);
        s3.add(new Month(3, 2010), 512);
        s3.add(new Month(4, 2010), 589);
        s3.add(new Month(5, 2010), 359);
        s3.add(new Month(6, 2010), 402);
        final TimeSeriesCollection dataset = new TimeSeriesCollection();
        dataset.addSeries(s1);
        dataset.addSeries(s2);
        dataset.addSeries(s3);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中调用 getDataset()方法获取数据集合，然后使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        XYDataset dataset = getDataset();
        JFreeChart chart = ChartFactory.createTimeSeriesChart("2010 年上半年销售量",  //图表标题
                "月份",                                                     //X 轴标签
                "销售量（单位：本）",                                         //Y 轴标签
                dataset,                                                    //数据集
                true,                                                       //是否显示图例
                false,                                                      //是否生成工具栏提示
                false                                                       //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中获取一个 XYPlot 实例，同时设置刻度单位。代码如下：

```
public static void updatePlot(JFreeChart chart) {
        //分类图表
        XYPlot xyPlot = chart.getXYPlot();
        //设置单位
        final DateAxis axis = (DateAxis) xyPlot.getDomainAxis();
        //设置刻度单位
        axis.setTickUnit(new DateTickUnit(DateTickUnitType.MONTH, 2,
                    new SimpleDateFormat("yyyy-MMM")));

}
```

秘笈心法

心法领悟 261：创建时间轴刻度。

使用 DateTickUnit 的构造方法可以创建时间轴刻度的实例。语法如下：

```
DateTickUnit(DateTickUnitType unitType, int multiple, DateFormat formatter)
```

参数说明

❶ unitType：表示刻度使用的单位类型，定义在 DateTickUnitType 常量中，如 DateTickUnitType.YEAR 表示年，DateTickUnitType.MONTH 表示月等。

❷ multiple：表示单位类型的跨度。

❸ formatter：表示显示的日期格式。

实例 262 设置时间轴范围

光盘位置：光盘\MR\09\262

高级

实用指数：★★★★

实例说明

绘制 JFreeChart 时序图时，可以根据需要设置 X 轴的取值范围。本实例将演示如何设置 X 轴的范围，运行结果如图 9.30 所示。

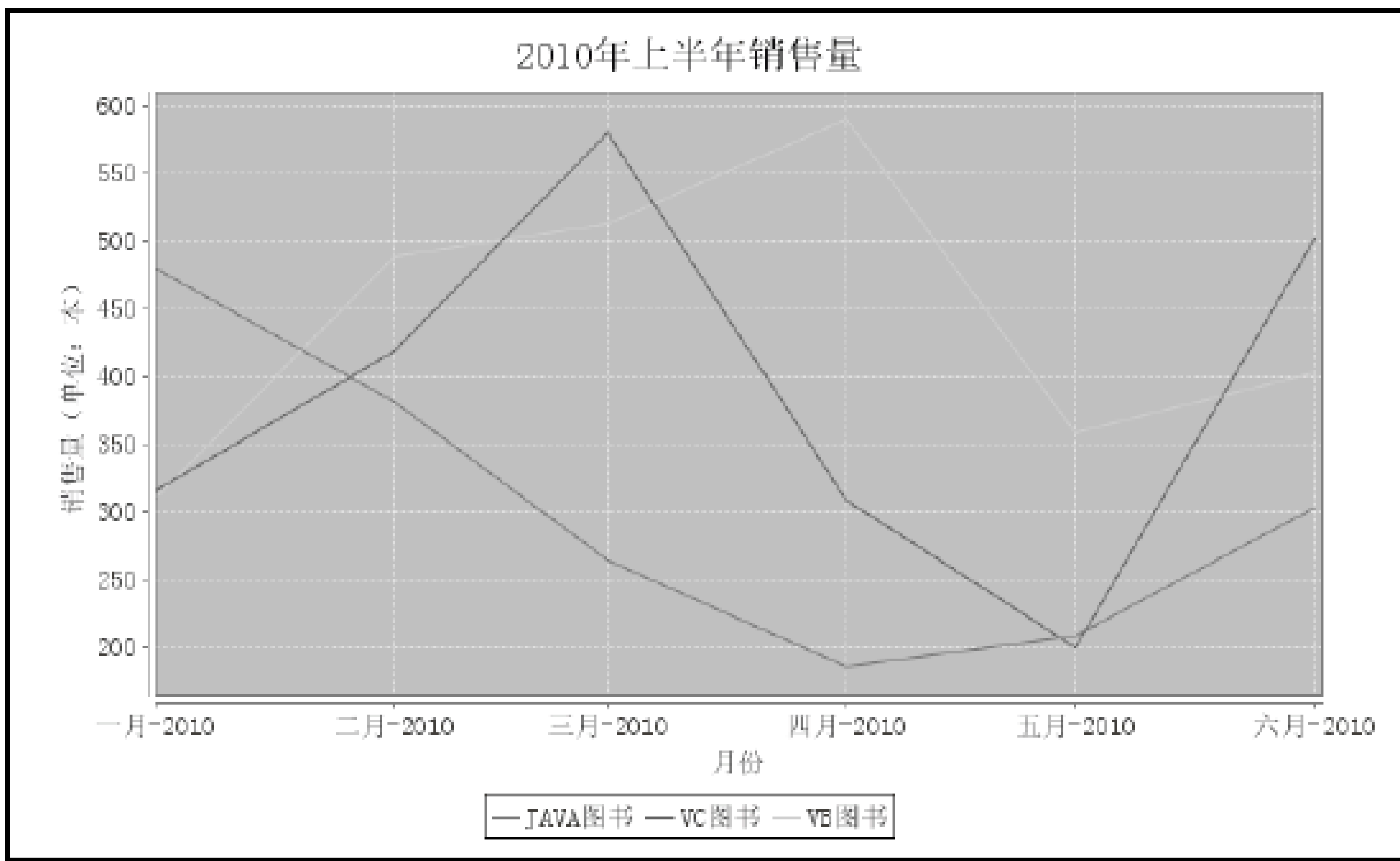


图 9.30　设置时间轴范围

关键技术

使用 DateAxis 类的 setRange()方法可以设置 X 轴的显示范围。语法如下：

```
setRange(Date lower, Date upper)
```

参数说明

❶ lower：表示 X 轴的开始时间。

❷ upper：表示 X 轴的结束时间。

（1）创建 getDatasetz()方法，在该方法中使用 TimeSeriesCollection 类创建一个数据集。代码如下：

```
public static XYDataset getDataset() {
        final TimeSeries s1 = new TimeSeries("JAVA 图书");
        s1.add(new Month(1, 2010), 480);
        s1.add(new Month(2, 2010), 381);
        s1.add(new Month(3, 2010), 264);
        s1.add(new Month(4, 2010), 185);
        s1.add(new Month(5, 2010), 209);
        s1.add(new Month(6, 2010), 302);
        final TimeSeries s2 = new TimeSeries("VC 图书");
        s2.add(new Month(1, 2010), 315);
```

```
        s2.add(new Month(2, 2010), 418);
        s2.add(new Month(3, 2010), 580);
        s2.add(new Month(4, 2010), 308);
        s2.add(new Month(5, 2010), 200);
        s2.add(new Month(6, 2010), 501);
        final TimeSeries s3 = new TimeSeries("VB 图书");
        s3.add(new Month(1, 2010), 310);
        s3.add(new Month(2, 2010), 489);
        s3.add(new Month(3, 2010), 512);
        s3.add(new Month(4, 2010), 589);
        s3.add(new Month(5, 2010), 359);
        s3.add(new Month(6, 2010), 402);
        final TimeSeriesCollection dataset = new TimeSeriesCollection();
        dataset.addSeries(s1);
        dataset.addSeries(s2);
        dataset.addSeries(s3);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中调用 getDataset()方法获取数据集合，然后使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        XYDataset dataset = getDataset();
        JFreeChart chart = ChartFactory.createTimeSeriesChart("2010 年上半年销售量",  //图表标题
                "月份",                                                  //X 轴标签
                "销售量（单位：本）",                                     //Y 轴标签
                dataset,                                                 //数据集
                true,                                                    //是否显示图例
                false,                                                   //是否生成工具栏提示
                false                                                    //是否生成 URL 链接
        );
        return chart;
}
```

（3）创建 updatePlot()方法，在该方法中创建两个 Day 类型变量，设置 X 轴的时间范围。代码如下：

```
private void updatePlot(JFreeChart chart) {
        Day endDay = new Day(1,6,2010);
                SerialDate serialDate = SerialDate.addMonths(-5, endDay.getSerialDate());
        Day beginDay = new Day(serialDate.toDate());
        DateAxis axis = (DateAxis) chart.getXYPlot().getDomainAxis();
        //设置 X 轴的开始、结束位置
        axis.setRange(beginDay.getStart(), endDay.getEnd());
}
```

秘笈心法

心法领悟 262：对月进行加、减法计算。

通过 SerialDate 类的 addMonths()方法可以对月进行加、减法计算。语法如下：

```
addMonths(int months, SerialDate base)
```

参数说明

❶ months：需要增加或者减少的月数，如果 months 为正数，则增加月数；如果 months 为负数，则减少月数。

❷ base：表示需要进行计算的日期。

9.4　联合分类图

实例 263　生成线形图与柱形图

高级

光盘位置：光盘\MR\09\263

实用指数：★★★★★

实例说明

联合分类图由多张图表共同组成，通过不同的图表展示同一个数据集。本实例将演示如何生成线形图与柱

形图的联合分类图，运行结果如图 9.31 所示。

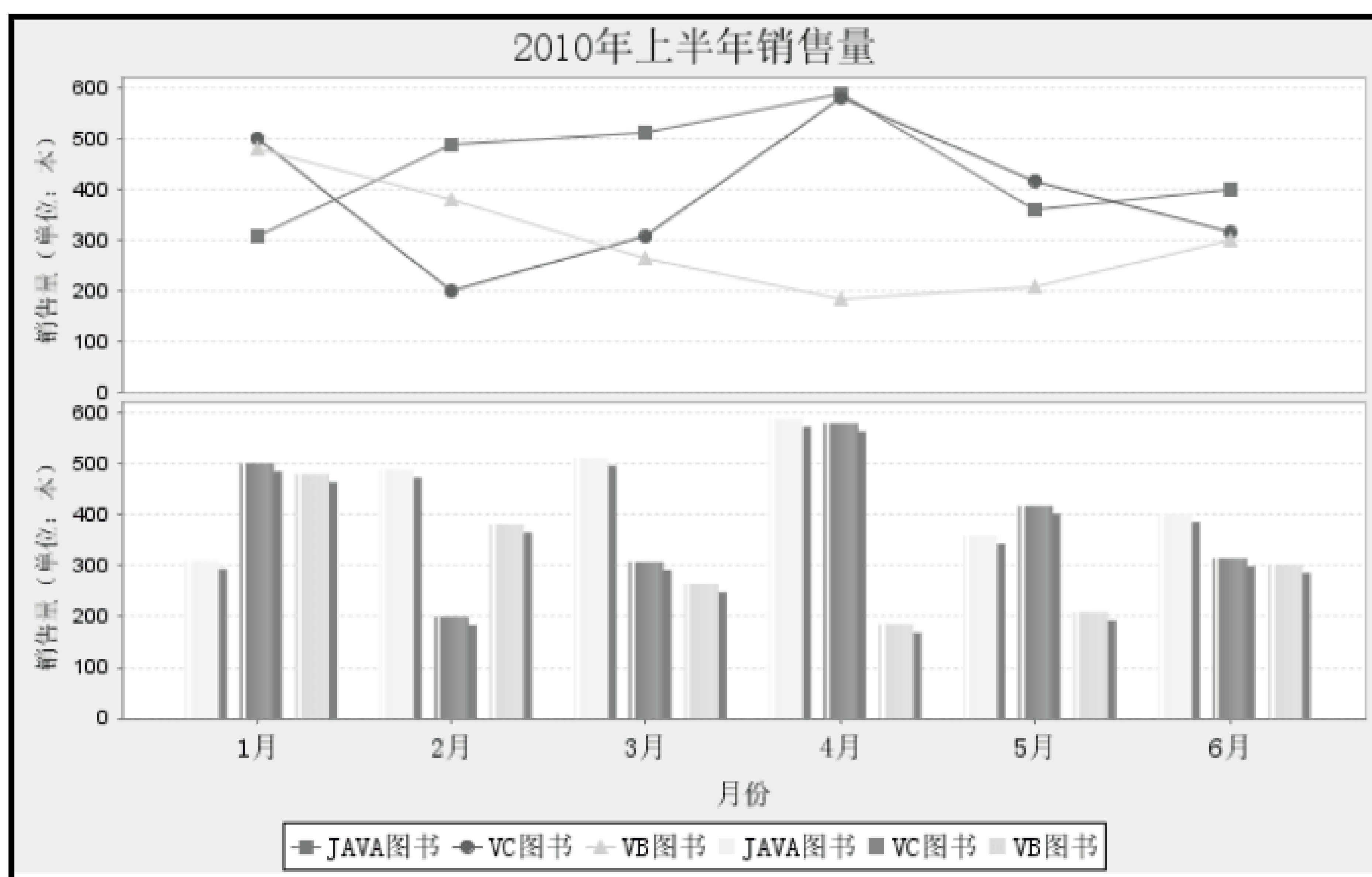


图 9.31　生成线形图与柱形图

关键技术

通过 CombinedDomainCategoryPlot 类可以使一个图表展示多个图形，利用该类的 add()方法可以添加多个 Plot。语法如下：

```
add(CategoryPlot subplot)
```

参数说明

subplot：表示需要添加的 Plot 实例。

（1）创建 getCategoryDataset()方法，在该方法中使用 DefaultCategoryDataset 类创建数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //行关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
```

```
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中创建 LineAndShapeRenderer 和 BarRenderer 实例，然后分别使用这两个实例渲染生成两个 Plot，再把这两个 Plot 添加到 CombinedDomainCategoryPlot 实例中，最后生成 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        //生成线形图渲染
        LineAndShapeRenderer renderer1 = new LineAndShapeRenderer();
        //生成柱形图渲染
        BarRenderer renderer2 = new BarRenderer();
        //设置 X 轴
        CategoryAxis domainAxis = new CategoryAxis("月份");
        //设置 Y 轴
        NumberAxis rangeAxis = new NumberAxis("销售量（单位：本）");
        //设置图表
        CategoryPlot plot1 = new CategoryPlot(dataset, domainAxis, rangeAxis, renderer1);
        CategoryPlot plot2 = new CategoryPlot(dataset, domainAxis, rangeAxis, renderer2);
        //设置联合分类图
        final CombinedDomainCategoryPlot plot = new CombinedDomainCategoryPlot(domainAxis);
        plot.add(plot1);
        plot.add(plot2);
        JFreeChart chart = new JFreeChart("2010 年上半年销售量", plot);
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改图表标题、X 轴、X 轴标签等字体。代码如下：

```
public static void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

秘笈心法

心法领悟 263：创建联合分类图。

使用 CombinedDomainCategoryPlot 类的构造方法可以创建联合分类图的实例。语法如下：

```
CombinedDomainCategoryPlot(CategoryAxis domainAxis)
```

参数说明

domainAxis：表示联合分类图中的 X 分类轴，在联合分类图中多个图表共用一个分类轴。

实例 264	设置图表高度 光盘位置：光盘\MR\09\264	高级 实用指数：★★★★

实例说明

本实例分为上、下两个图表，默认情况下其高度一致，在此根据实际需要修改指定图表默认高度，运行结果如图 9.32 所示。

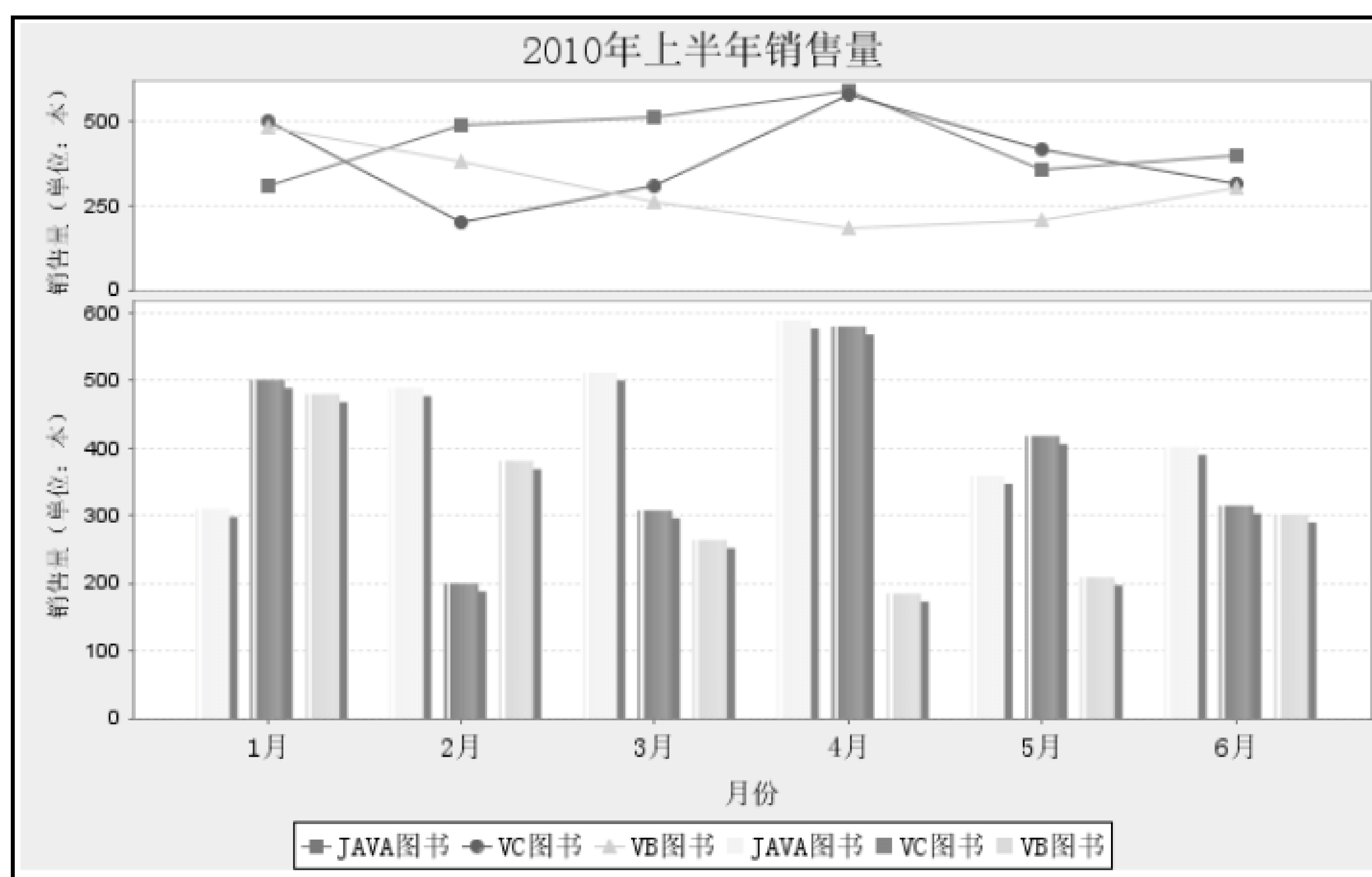


图 9.32　生成线形图与柱形图

■ 关键技术

利用 CombinedDomainCategoryPlot 类的 add()方法可以添加多个 Plot，还可以设置指定图表的高度。语法如下：

```
add(CategoryPlot subplot, int weight)
```

参数说明

❶ subplot：表示准备添加的 Plot 实例。

❷ weight：表示添加的 Plot 实例的高度。

■

（1）创建 getCategoryDataset()方法，在该方法中使用 DefaultCategoryDataset 类创建数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //行关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
```

```
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);

        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中创建 LineAndShapeRenderer 和 BarRenderer 实例，然后分别使用这两个实例渲染生成两个 Plot，再把这两个 Plot 添加到 CombinedDomainCategoryPlot 实例中，最后生成 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        //生成线形图渲染
        LineAndShapeRenderer renderer1 = new LineAndShapeRenderer();
        //生成柱形图渲染
        BarRenderer renderer2 = new BarRenderer();
        //设置 X 轴
        CategoryAxis domainAxis = new CategoryAxis("月份");
        //设置 Y 轴
        NumberAxis rangeAxis = new NumberAxis("销售量（单位：本）");
        //设置图表
        CategoryPlot plot1 = new CategoryPlot(dataset, domainAxis, rangeAxis, renderer1);
        CategoryPlot plot2 = new CategoryPlot(dataset, domainAxis, rangeAxis, renderer2);
        //设置联合分类图
        final CombinedDomainCategoryPlot plot = new CombinedDomainCategoryPlot(domainAxis);
        plot.add(plot1, 1);
        plot.add(plot2, 2);
        JFreeChart chart = new JFreeChart("2010 年上半年销售量", plot);
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改图表标题、X 轴标签、X 轴标签等字体。代码如下：

```
public static void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

秘笈心法

心法领悟 264：设置图表高度。

向 CombinedDomainCategoryPlot 类中添加图表示例时，设置的高度数值是总图表高度的比例值。例如，线形图的高度设置为 1，柱形图的高度设置为 2 时，表示图表的总高度为 3，那么线形图的高度占总图表的 1/3，柱形图的高度占总图表的 2/3。

实例 265	设置图表位置 光盘位置：光盘\MR\09\265	高级 实用指数：★★★★

实例说明

联合分类图中有多个图表，需要根据实际情况设置其显示位置。本实例把柱形图显示在图表上面，把线形图显示在图表下面，运行结果如图 9.33 所示。

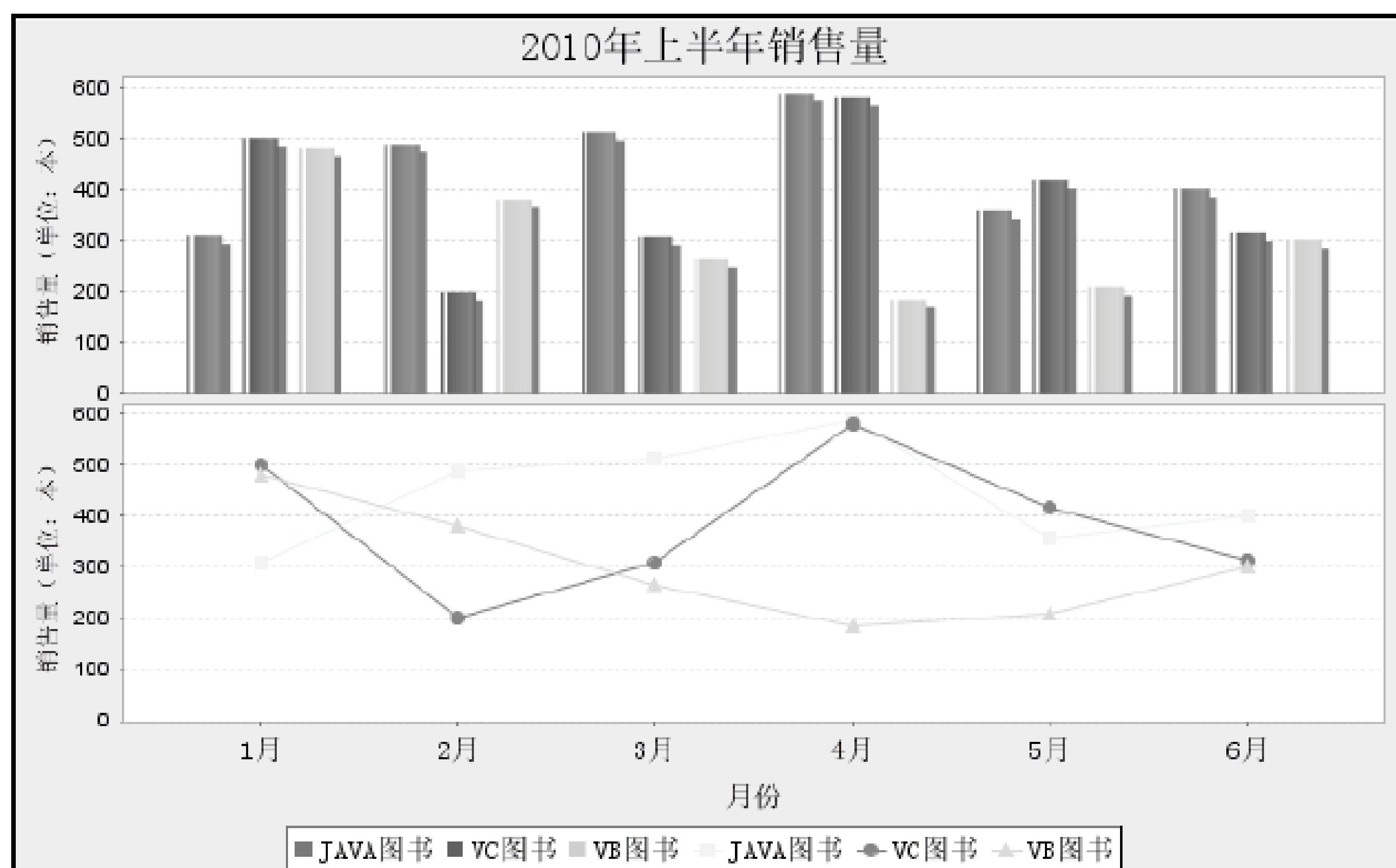


图 9.33　生成线形图与柱形图

关键技术

图表的显示顺序与添加 Plot 实例的顺序一致，先添加的 Plot 实例显示在图表上面，后添加的 Plot 实例显示在图表下面。代码如下：

```
final CombinedDomainCategoryPlot plot = new CombinedDomainCategoryPlot(domainAxis);
//设置联合图表
plot.add(plot2, 1);
plot.add(plot1, 1);
```

（1）创建 getCategoryDataset()方法，在该方法中使用 DefaultCategoryDataset 类创建数据集。代码如下：

```
private static CategoryDataset getCategoryDataset() {
        //行关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（2）创建 getJFreeChart()方法，在该方法中创建 LineAndShapeRenderer 和 BarRenderer 实例，然后分别使用这两个实例生成两个 Plot，再把这两个 Plot 添加到 CombinedDomainCategoryPlot 实例中，最后生成 JFreeChart 对象。代码如下：

```
public static JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        //生成线形图渲染
        LineAndShapeRenderer renderer1 = new LineAndShapeRenderer();
        //生成柱形图渲染
        BarRenderer renderer2 = new BarRenderer();
        //设置 X 轴
        CategoryAxis domainAxis = new CategoryAxis("月份");
        //设置 Y 轴
        NumberAxis rangeAxis = new NumberAxis("销售量（单位：本）");
        //设置图表
        CategoryPlot plot1 = new CategoryPlot(dataset, domainAxis, rangeAxis, renderer1);
        CategoryPlot plot2 = new CategoryPlot(dataset, domainAxis, rangeAxis, renderer2);
        //设置联合分类图
        final CombinedDomainCategoryPlot plot = new CombinedDomainCategoryPlot(domainAxis);
        plot.add(plot2, 1);
        plot.add(plot1, 1);
        JFreeChart chart = new JFreeChart("2010 年上半年销售量", plot);
        return chart;
}
```

（3）创建 updateFont()方法，在该方法中修改图表标题、X 轴标签、X 轴标签等字体。代码如下：

```
public static void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        // X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        // X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

秘笈心法

心法领悟 265：使用的渲染类型决定绘制图表的类型。

绘制联合分类图时，要把多个 CategoryPlot 实例添加到 CombinedDomainCategoryPlot 类中。创建 CategoryPlot 实例时使用的渲染类型决定了绘制图表的类型。本实例创建了线形图与柱形图，所以必须使用 LineAndShapeRenderer 和 BarRenderer 类初始化 CategoryPlot。

9.5　图表的综合应用

实例 266	利用饼图分析不同编程语言的市场占有率 光盘位置：光盘\MR\09\266	高级 实用指数：★★★★

学习了饼图的绘制方法之后，下面介绍如何利用 JFreeChart 组件生成的饼图显示不同编程语言的市场占有率情况。程序运行结果如图 9.34 所示。

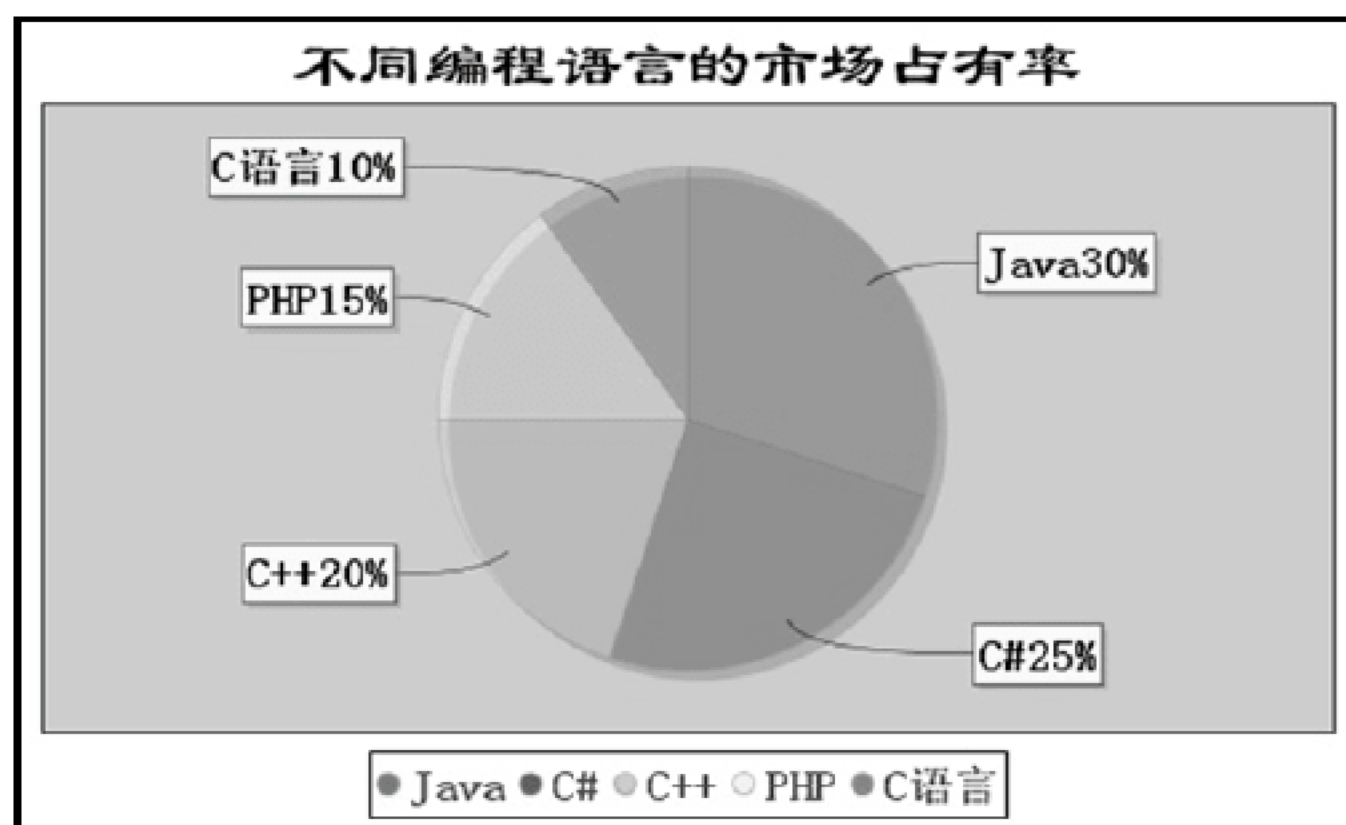


图 9.34　不同编程语言的市场占有率情况

关键技术

在制图前，首先创建主题样式并指定样式中的字体。可以通过 ChartFactory 的 setChartTheme()方法设置主题样式。在指定制图样式后，ChartFactory 对象创建的制图对象将按此样式进行显示，图表中的文字将按指定的字体进行显示。例如：

```
StandardChartTheme standardChartTheme = new StandardChartTheme("CN");          //创建主题样式
standardChartTheme.setExtraLargeFont(new Font("隶书", Font.BOLD, 20));         //设置标题字体
standardChartTheme.setRegularFont(new Font("宋体", Font.PLAIN, 15));           //设置图例的字体
standardChartTheme.setLargeFont(new Font("宋体", Font.PLAIN, 15));             //设置轴向的字体
ChartFactory.setChartTheme(standardChartTheme);                               //应用主题样式
```

通过上述代码设置主题样式后，再通过 ChartFactory 创建 JFreeChart 的对象，可以解决中文乱码问题。

（1）创建 ChartUtil 类，在该类中编写生成饼图数据集合的方法 getDataset()。关键代码如下：

```
private static PieDataset getDataset(){
        DefaultPieDataset dataset = new DefaultPieDataset();                   //创建饼图数据集合
        dataset.setValue("Java", 30);                                         //添加数据
        dataset.setValue("C#", 25);                                           //添加数据
        dataset.setValue("C++", 20);                                          //添加数据
        dataset.setValue("PHP", 15);                                          //添加数据
        dataset.setValue("C 语言", 10);                                       //添加数据
        return dataset;
}
```

（2）编写生成饼图的方法 createChart()。关键代码如下：

```
public static JFreeChart createChart(){
        StandardChartTheme standardChartTheme = new StandardChartTheme("CN");  //创建制图的主题样式
        standardChartTheme.setLargeFont(new Font("黑体", Font.BOLD, 16));      //设置轴向的字体
        standardChartTheme.setRegularFont(new Font("宋体", Font.BOLD, 16));    //设置图例的字体
        standardChartTheme.setExtraLargeFont(new Font("隶书", Font.BOLD, 24)); //设置标题字体
        ChartFactory.setChartTheme(standardChartTheme);                       //设置制图工厂使用主题
        JFreeChart chart = ChartFactory.createPieChart(
                                "不同编程语言的市场占有率",                     //图表标题
                                getDataset(),                                 //绘制数据
                                true,                                         //定义图表是否包含图例
                                true,                                         //定义图表是否包含提示
                                false);                                       //定义图表是否包含 URL
        PiePlot plot = (PiePlot)chart.getPlot();
        plot.setLabelGenerator(
                        new StandardPieSectionLabelGenerator("{0}{2}",
                        NumberFormat.getNumberInstance(),
                        NumberFormat.getPercentInstance()));                  //设置分类标签的格式，更改数字的显示格式为百分比
        plot.setBackgroundAlpha(0.8f);                                        //设置背景透明度
        plot.setForegroundAlpha(0.4f);                                        //设置前景透明度
        return chart;
}
```

（3）创建 index.jsp 页，显示 JFreeChart 组件生成的饼图。具体代码参见配书光盘。

秘笈心法

心法领悟 266：抽取分类图形。

应用表示饼图数据区域的 PiePlot 对象的 setExplodePercent()方法，可以将饼图中表示 C#的模块抽离出来。

实例 267 利用柱形图显示某 Ajax 网站不同框架的年下载量

光盘位置：光盘\MR\09\267

高级

实用指数：★★★★

实例说明

本实例将介绍如何应用 JFreeChart 生成的柱形图显示某 Ajax 网站不同框架的年下载量，运行结果如图 9.35 所示。由于这些框架的下载量数据都是笔者随机生成的，所以并不能保证其准确性。

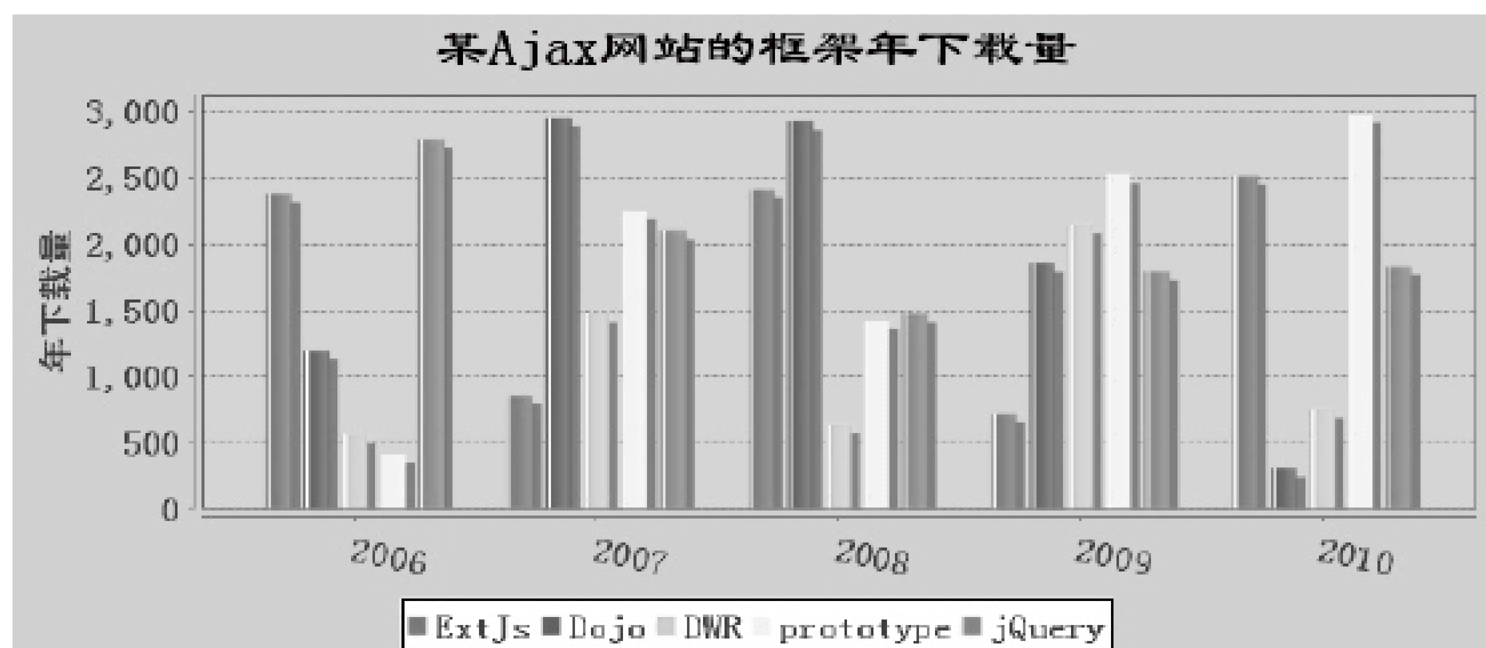


图 9.35 利用柱形图显示某 Ajax 网站不同框架的年下载量

关键技术

在本实例中，设置了 X 轴文字显示样式为倾斜。可以通过 CategoryPlot 对象的 getDomainAxis()方法获取表示 X 轴的对象 CategoryAxis，然后通过 CategoryAxis 对象的 setCategoryLabelPositions()方法设置文字的旋转。

（1）创建 ChartUtil 类，编写用于填充柱形图数据集合的方法 getDataset()。关键代码如下：

```
private static CategoryDataset getDataset(){
    DefaultCategoryDataset dataset = new DefaultCategoryDataset();
    for(int i=2006;i<=2010;i++){
        dataset.addValue(new Random().nextInt(3000), "ExtJs", i+"");
        dataset.addValue(new Random().nextInt(3000), "Dojo", i+"");
        dataset.addValue(new Random().nextInt(3000), "DWR", i+"");
        dataset.addValue(new Random().nextInt(3000), "prototype", i+"");
        dataset.addValue(new Random().nextInt(3000), "jQuery", i+"");
    }
    return dataset;
}
```

（2）编写绘制柱形图的方法 createChart()。代码如下：

```
public static JFreeChart createChart(){
    StandardChartTheme standardChartTheme = new StandardChartTheme("CN");          //创建制图的主题样式
    standardChartTheme.setLargeFont(new Font("黑体", Font.BOLD, 16));              //设置轴向的字体
    standardChartTheme.setRegularFont(new Font("宋体", Font.BOLD, 16));            //设置图例的字体
    standardChartTheme.setExtraLargeFont(new Font("隶书", Font.BOLD, 24));         //设置标题字体
    ChartFactory.setChartTheme(standardChartTheme);                                //设置制图工厂使用主题
    //创建效果图
    JFreeChart chart = ChartFactory.createBarChart(
                        "某 Ajax 网站的框架年下载量",                              //图表标题
                        "",                                                        //坐标标题
                        "年下载量",                                                //坐标标题
                        getDataset(),                                              //绘制数据
                        PlotOrientation.VERTICAL,                                  //直方图的方向：竖向
                        true,                                                      //定义图表是否包含图例
                        false,                                                     //定义图表是否包含提示
                        false);                                                    //定义图表是否包含 URL
        chart.setBackgroundPaint(new Color(168, 219, 219));
        CategoryPlot plot = chart.getCategoryPlot();
        plot.setBackgroundPaint(new Color(219, 219, 127));                         //设置绘图区域背景色
        plot.setDomainGridlinesVisible(false);                                     //设置垂直方向网格线是否显示
```

```
        plot.setRangeGridlinePaint(Color.RED);                          //设置水平方向网格线的颜色
        plot.setRangeGridlinesVisible(true);                            //设置水平方向网格线是否显示
        CategoryAxis domainAxis = (CategoryAxis) plot.getDomainAxis();  //X 轴对象
        domainAxis.setCategoryLabelPositions(
                CategoryLabelPositions.createDownRotationLabelPositions(Math.PI /   //文字顺时针旋转
                16.0) );                                                //文字旋转弧度，接受双精度参数
        return chart;
}
```

（3）创建 index.jsp 页面，显示 JFreeChart 生成的图表。具体代码参见配书光盘。

秘笈心法

心法领悟 267：设置图表网格线的颜色。

可以通过 CategoryPlot 对象的 setRangeGridlinePaint()方法设置水平方向网格线的颜色，通过 setDomainGridlinePaint()方法可以设置垂直方向网格线的颜色。

实例 268　利用折线图分析不同城市气温变化情况　高级

光盘位置：光盘\MR\09\268　实用指数：★★★★

实例说明

使用折线图可以更清晰地查看统计结果的变化情况。本例将通过折线图显示 3—11 月不同城市的气温变化情况，运行结果如图 9.36 所示。

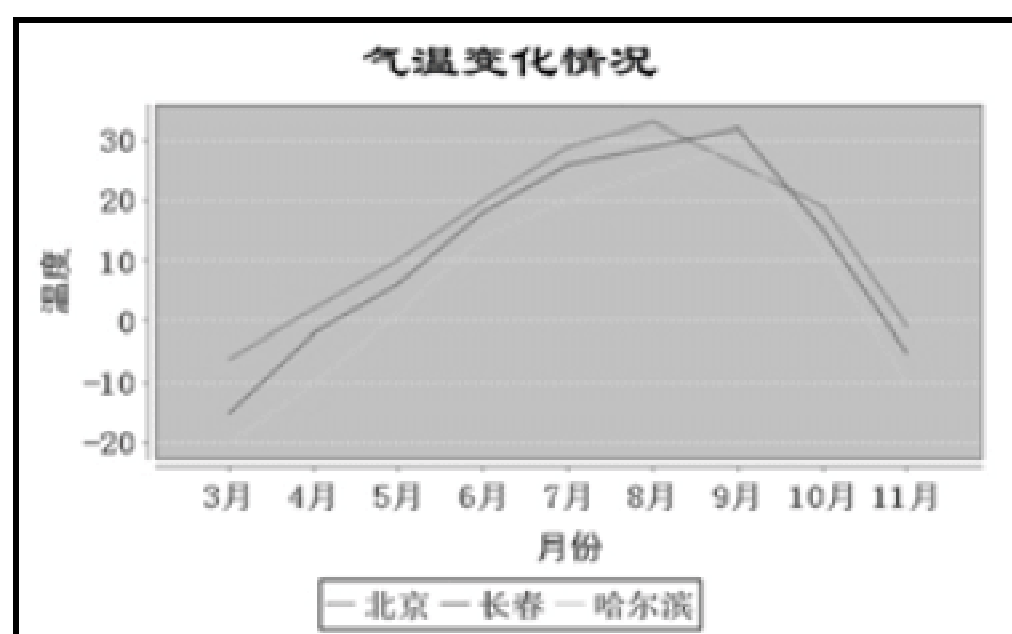


图 9.36　反映不同城市气温变化情况的折线图

关键技术

可以应用 DefaultCategoryDataset 作为折线图的数据集合，然后通过 addValue()方法向数据集中添加数据。addValue()方法的语法如下：

```
public void addValue(Number value, Comparable rowKey, Comparable columnKey)
```

参数说明

❶ value：指定要添加的数据。

❷ rowKey：指定要添加数据的是哪一个折线对象。

❸ columnKey：指定在哪一列上添加数据，X 轴的不同刻度值代表不同的列。

（1）创建 ChartUtil 类，编写用于生成折线图数据集合的方法 getDataset()。关键代码如下：

```
private static CategoryDataset getDataset(){
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();      //创建数据集合
        Number[]temperature1 = {-6,2,10,20,29,33,26,19,-1};                 //不同月份的温度值数组
        Number[]temperature2 = {-15,-2,6,18,26,29,32,15,-5};
        Number[]temperature3 = {-20,-10,1,14,20,25,29,12,-10};
        for(int i=3;i<=11;i++){
                dataset.addValue(temperature1[i-3], "北京", i+"月");       //添加数据
                dataset.addValue(temperature2[i-3], "长春", i+"月");       //添加数据
```

```
            dataset.addValue(temperature3[i-3], "哈尔滨", i+"月");                //添加数据
        }
        return dataset;
}
```

（2）编写绘制折线图的方法 createChart()，关键代码如下。

```
public static JFreeChart createChart(){
        StandardChartTheme standardChartTheme = new StandardChartTheme("CN");    //创建制图的主题样式
        standardChartTheme.setLargeFont(new Font("黑体", Font.BOLD, 16));        //设置轴向的字体
        standardChartTheme.setRegularFont(new Font("宋体", Font.BOLD, 16));      //设置图例的字体
        standardChartTheme.setExtraLargeFont(new Font("隶书", Font.BOLD, 24));   //设置标题字体
        ChartFactory.setChartTheme(standardChartTheme);                          //设置制图工厂使用主题
        JFreeChart chart = ChartFactory.createLineChart(
                                "气温变化情况",                                   //图表标题
                                "月份",
                                "温度",
                                getDataset(),                                    //绘制数据
                                PlotOrientation.VERTICAL,                        //图表方向
                                true,                                            //定义图表是否包含图例
                                false,                                           //定义图表是否包含提示
                                false);                                          //定义图表是否包含 URL
          return chart;
}
```

秘笈心法

心法领悟 268：加粗折线。

通过 LineAndShapeRenderer 类的 setSeriesStroke()方法可以设置指定折线的笔触。设置折线笔触之前，首先需要利用 CategoryPlot 对象的 CategoryPlot()方法获取到 LineAndShapeRenderer 对象。setSeriesStroke()方法的语法如下：

```
public void setSeriesStroke(int series,Stroke stroke)
```

参数说明

❶ series：指定要设置笔触的折线对象的索引。

❷ stroke：BasicStroke 类型的对象，指定折线的笔触。

实例 269　利用区域图分析不同学生的成绩变化

高级

光盘位置：光盘\MR\09\269

实用指数：★★★★

实例说明

区域图的表现形式是多样的，可用于描述数据的变化程度、部分与整体的关系、数据对比等。本例将通过不同学生的成绩数据绘制区域图，对比分析学生的成绩变化，运行结果如图 9.37 所示。

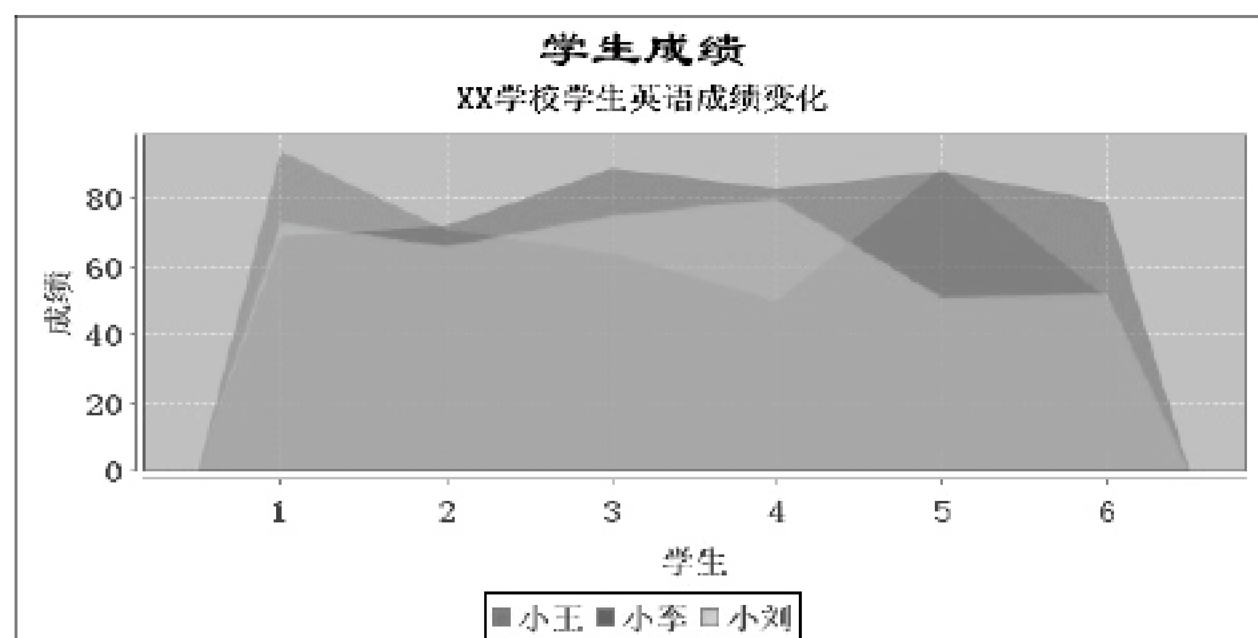


图 9.37　学生成绩的区域图

关键技术

通过 TextTitle 类可以为图表添加子标题，可以调用该类的 setVerticalAlignment()方法设置子标题的显示位置，然后通过 JFreeChart 对象的 addSubtitle(Title title)方法设置此子标题的内容。

（1）创建 AreaChartUtil 类，编写获取区域图数据集合的方法。代码如下：

```
public static CategoryDataset createDataset() {
	//数据集合 DefaultCategoryDataset 对象
	DefaultCategoryDataset defaultcategorydataset = new DefaultCategoryDataset();
	Random random = new Random();                                          //创建 Random 对象
	//向数据集合加入 6 个月的数据
	for (int i = 1; i < 7; i++) {
		defaultcategorydataset.addValue(random.nextInt(50) + 50, "小王", i + "");
		defaultcategorydataset.addValue(random.nextInt(50) + 50, "小李", i + "");
		defaultcategorydataset.addValue(random.nextInt(50) + 50, "小刘", i + "");
	}
	return defaultcategorydataset;
}
```

（2）编写 createChart()方法生成区域图。代码如下：

```
public static JFreeChart createChart() {
	JFreeChart jfreechart = null;                                          //JFreeChart 对象
	StandardChartTheme standardChartTheme = new StandardChartTheme("CN");
	standardChartTheme.setExtraLargeFont(new Font("隶书", Font.BOLD, 24));   //设置标题字体
	standardChartTheme.setRegularFont(new Font("宋体", Font.BOLD, 15));      //设置图例的字体
	standardChartTheme.setLargeFont(new Font("宋体", Font.BOLD, 15));        //设置轴向的字体
	ChartFactory.setChartTheme(standardChartTheme);                        //设置主题样式
	jfreechart=ChartFactory.createAreaChart(
			"学生成绩",                                                    //图表标题
			"学生",                                                        //横轴标题
			"成绩",                                                        //纵轴标题
			createDataset(),                                               //制图的数据集
			PlotOrientation.VERTICAL,                                      //定义区域图的方向为纵向
			true,                                                          //是否显示图例标识
			true,                                                          //是否显示 tooltips
			false);                                                        //是否支持超链接
	//获取 CategoryPlot 对象
	CategoryPlot categoryplot = (CategoryPlot)jfreechart.getPlot();
	categoryplot.setForegroundAlpha(0.5F);                                 //设置前景透明度为 50%
	categoryplot.setDomainGridlinesVisible(true);                          //显示网格
	//创建子标题
	TextTitle textTile = new TextTitle("XX 学校学生英语成绩变化");
	textTile.setVerticalAlignment(VerticalAlignment.BOTTOM);               //居中显示
	jfreechart.addSubtitle(textTile);                                      //将子标题放入 JFreeChart 中
	//获取 NumberAxis 对象
	NumberAxis numberaxis = (NumberAxis)categoryplot.getRangeAxis();
	numberaxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits());
	ChartUtilities.applyCurrentTheme(jfreechart);
	return jfreechart;
}
```

秘笈心法

心法领悟 269：设置 Y 轴的数据类型。

通过 CategoryPlot 的 getRangeAxis()方法可获取 Y 轴的数据对象。该方法返回的是 ValueAxis 对象，本实例将其强制转换为 NumberAxis 对象，并通过 setStandardTickUnits()方法设置数据为整型。

实例 270	利用时序图分析股票价格走势 光盘位置：光盘\MR\09\270	高级 实用指数：★★★★

实例说明

时序图是一种数据与日期、时间联系非常紧密的图表，常用于金融和财经类的图表绘制，如股市的走势图、货币变动趋势图等。本例将利用时序图分析股票价格走势，运行结果如图 9.38 所示。

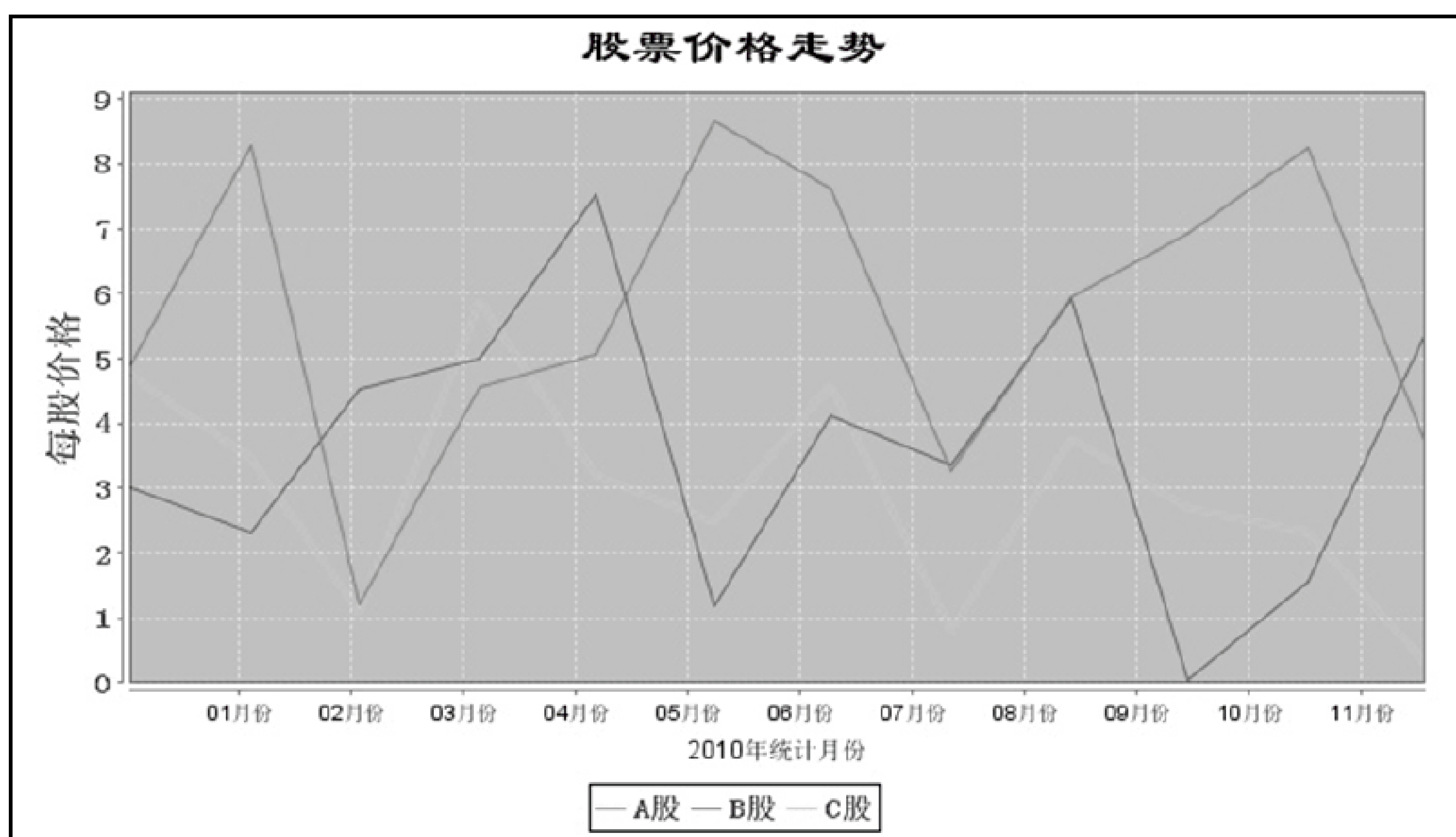


图 9.38　利用时序图分析股票价格走势

关键技术

本实例在为 X 轴添加标签数据时应用了 DateAxis 类，该类的对象用于设置日期时间类型的数据。在设置日期数据格式时，首先需要创建 SimpleDateFormat 对象，表示日期时间格式器；然后创建 DateTickUnit 对象，并将 SimpleDateFormat 对象作为 DateTickUnit 构造方法参数；接下来，通过 DateAxis 类的 setTickUnit()方法应用设置的日期格式；最后调用 XYPlot 对象的 setDomainAxis()方法设置为图表的 X 轴刻度值。关键代码如下：

```
XYPlot plot = timeChart.getXYPlot();
DateFormat format = new SimpleDateFormat("MM 月份");                    //创建日期格式对象
DateAxis domainAxis = new DateAxis("2010 年统计月份");                  //创建时间轴对象
DateTickUnit dtu = new DateTickUnit(DateTickUnit.DAY, 29, format);
domainAxis.setTickUnit(dtu);                                            //设置横轴上的时间刻度的显示格式
plot.setDomainAxis(domainAxis);                                         //为绘图属性添加横轴对象
```

（1）创建 TimeSeriesUtil 类，编写创建时序图数据集合的方法 getDataset()。时序图的数据集合为 TimeSeriesCollection 类型，其数据为 org.jfree.data.time.TimeSeries 对象类型。通过 TimeSeriesCollection 类实例对象的 addSeries(TimeSeries timeseries)方法可以向数据集合中添加 TimeSeries 类型的数据。关键代码如下：

```
private static XYDataset getDataset(){
        TimeSeriesCollection dataset = new TimeSeriesCollection();      //创建时序图的数据集合
        TimeSeries timeSeriesA = new TimeSeries("A 股");                //A 股数据对象
        TimeSeries timeSeriesB = new TimeSeries("B 股");                //B 股数据对象
        TimeSeries timeSeriesC = new TimeSeries("C 股");                //C 股数据对象
        for(int i=1;i<=12;i++){                                         //循环一年的月份
                timeSeriesA.add(new Month(i,2010),new Random().nextDouble()*9);   //向 A 股对象中添加随机数据
                timeSeriesB.add(new Month(i,2010),new Random().nextDouble()*8);   //向 B 股对象中添加随机数据
                timeSeriesC.add(new Month(i,2010),new Random().nextDouble()*6);   //向 C 股对象中添加随机数据
        }
        dataset.addSeries(timeSeriesA);                                 //将数据对象添加至数据集合
        dataset.addSeries(timeSeriesB);                                 //将数据对象添加至数据集合
        dataset.addSeries(timeSeriesC);                                 //将数据对象添加至数据集合
        return dataset;
}
```

（2）编写生成时序图的方法 getTimeSeriesChart()，调用 CharFactory 的 createTimeSeriesChart()方法创建时序图对象。关键代码如下：

```
public static JFreeChart getTimeSeriesChart(){
        StandardChartTheme standardChartTheme = new StandardChartTheme("CN");
        standardChartTheme.setExtraLargeFont(new Font("隶书", Font.BOLD, 24));    //设置标题字体
        standardChartTheme.setRegularFont(new Font("宋体", Font.BOLD, 14));       //设置图例的字体
```

```
        standardChartTheme.setLargeFont(new Font("宋体", Font.BOLD, 18));           //设置轴向的字体
        ChartFactory.setChartTheme(standardChartTheme);                          //设置主题样式
        JFreeChart timeChart = ChartFactory.createTimeSeriesChart(
                    "股票价格走势",
                    "月份",
                    "每股价格",
                    getDataset(),
                    true,
                    true,
                    false);
        XYPlot plot = timeChart.getXYPlot();
        DateFormat format = new SimpleDateFormat("MM 月份");                      //创建日期格式对象
        DateAxis domainAxis = new DateAxis("2010 年统计月份");                     //创建时间轴对象
        DateTickUnit dtu = new DateTickUnit(DateTickUnit.DAY, 29, format);
        domainAxis.setTickUnit(dtu);                                             //设置横轴上的时间刻度的显示格式
        domainAxis.setLowerMargin(0.0);                                          //设置图表左侧边缘空白
        domainAxis.setUpperMargin(0.0);                                          //设置图表右侧边缘空白
        plot.setDomainAxis(domainAxis);                                          //为绘图属性添加横轴对象
        return timeChart;
}
```

秘笈心法

心法领悟 270：设置图表数据区的边缘间距。

DateAxis 对象的 setLowerMargin()方法用于设置图表数据区与图表左侧边缘（底部）的间距，setUpperMargin()方法用于设置图表数据区与图表右侧边缘（顶部）的间距。

实例 271	利用时序图分析 2009 年国际原油价格走势 光盘位置：光盘\MR\09\271	高级 实用指数：★★★★

实例说明

本实例介绍如何使用时序图分析 2009 年国际原油价格走势，运行结果如图 9.39 所示。程序中生成图表的数据是随机的，所以每次刷新页面时该时序图都会改变。

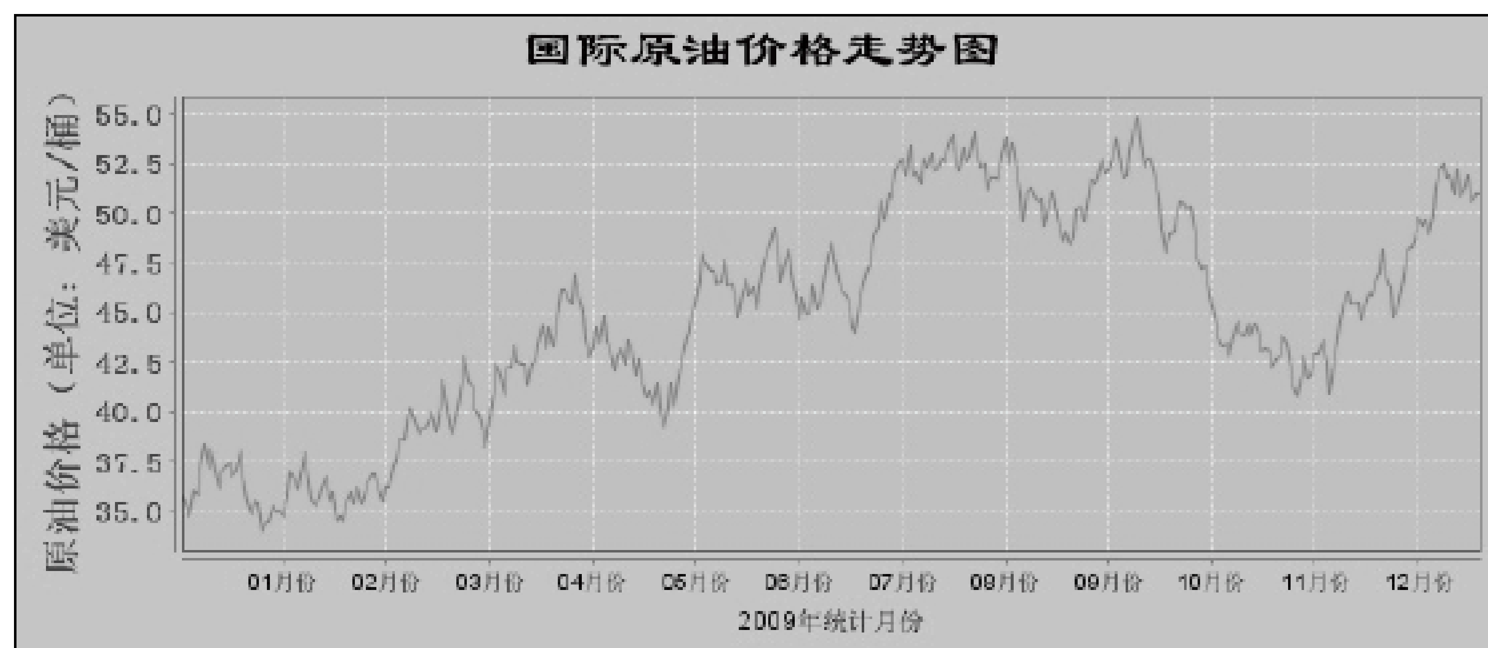


图 9.39　利用时序图分析 2009 年国际原油价格走势

关键技术

生成时序图数据集合时，需要创建 TimesSeries 类的对象。TimesSeries 是用来描述时间序列的一个数据集对象，通过该对象的 add()方法可以为数据集添加新的数据项。语法如下：

```
public void add(RegularTimePeriod period,double value)
```

参数说明

❶ period：添加到数据集的键名，它是一个日期或时间类型的对象。本实例中应用的是 JFreeChart 提供的 Day 对象，表示某一天的日期。

❷ value：要添加到数据集合的键值。本实例中添加的是随机的原油价格。

（1）创建 TimesSeriesUtil 类，编写生成时序图数据集合的方法 createDataset()。代码如下：

```
public static XYDataset createDataset() {
        TimeSeries timeseries = new TimeSeries("Random Data");
        Day day = new Day(1, 1, 2009);
```

```
        double value = 35;
        //添加一年 365 天的数据
        for (int i = 0; i < 365; i++) {
                double flag = Math.random();
                if(flag>0.5){
                        value=value+Math.random()*1.5;

                }else{
                        value=value-Math.random()*1.5;
                }
                timeseries.add(day, value);
                day = (Day) day.next();
        }
        //返回数据集合对象
        return new TimeSeriesCollection(timeseries);
}
```

（2）编写 createChart()方法，生成原油价格的时序图。代码如下：

```
public static JFreeChart createChart() {
        StandardChartTheme standardChartTheme = new StandardChartTheme("CN");
        standardChartTheme.setExtraLargeFont(new Font("隶书", Font.BOLD, 24));        //设置标题字体
        standardChartTheme.setRegularFont(new Font("宋体", Font.BOLD, 14));           //设置图例的字体
        standardChartTheme.setLargeFont(new Font("宋体", Font.BOLD, 18));             //设置轴向的字体
        ChartFactory.setChartTheme(standardChartTheme);                              //设置主题样式
        JFreeChart jfreechart = ChartFactory.createTimeSeriesChart(
                        "国际原油价格走势图",
                        "",
                        "原油价格（单位：美元/桶）",
                        createDataset(),
                        false,
                        false,
                        false);
        jfreechart.setBackgroundPaint(Color.ORANGE);                                 //设置背景色
        XYPlot xyplot = jfreechart.getXYPlot();                                      //获取图表的绘制属性
        DateFormat format = new SimpleDateFormat("MM 月份");                          //创建日期格式对象
        DateAxis domainAxis = new DateAxis("2009 年统计月份");                         //创建时间轴对象
        domainAxis.setDateFormatOverride(format);
        domainAxis.setLowerMargin(0.0);                                              //设置图表空白
        domainAxis.setUpperMargin(0.0);                                              //设置图表空白
        xyplot.setDomainAxis(domainAxis);                                            //为绘图属性添加横轴对象
        return jfreechart;
}
```

（3）创建 index.jsp 页，显示 JFreeChart 生成的时序图表。具体代码参见配书光盘。

秘笈心法

心法领悟 271：随机的原油价格取值。

在实现本实例时，可以定义一个表示原油价格的变量，取值为 35，然后通过 for 循环一年 365 天，随机生成 365 个大于或小于 35 的一系列的原油价格的浮点值，可用这些值作为时序图中每天的数据，这样就能显示出一年价格的波动情况。

实例 272	利用组合图表分析学生零用钱收支情况 光盘位置：光盘\MR\09\272	高级 实用指数：★★★★

实例说明

在实际应用中，常常需要显示两张或者两张以上图表组合而成的图表。组合图表一般分为共用 X 轴组合图表和共用 Y 轴组合图表。共用 X 轴组合图表中，两张图表共用 X 轴。如在本实例中，学生经济来源和学生经济支出图表的 X 轴都是学生姓名，而 Y 轴则是各自的 Y 轴，运行结果如图 9.40 和图 9.41 所示。

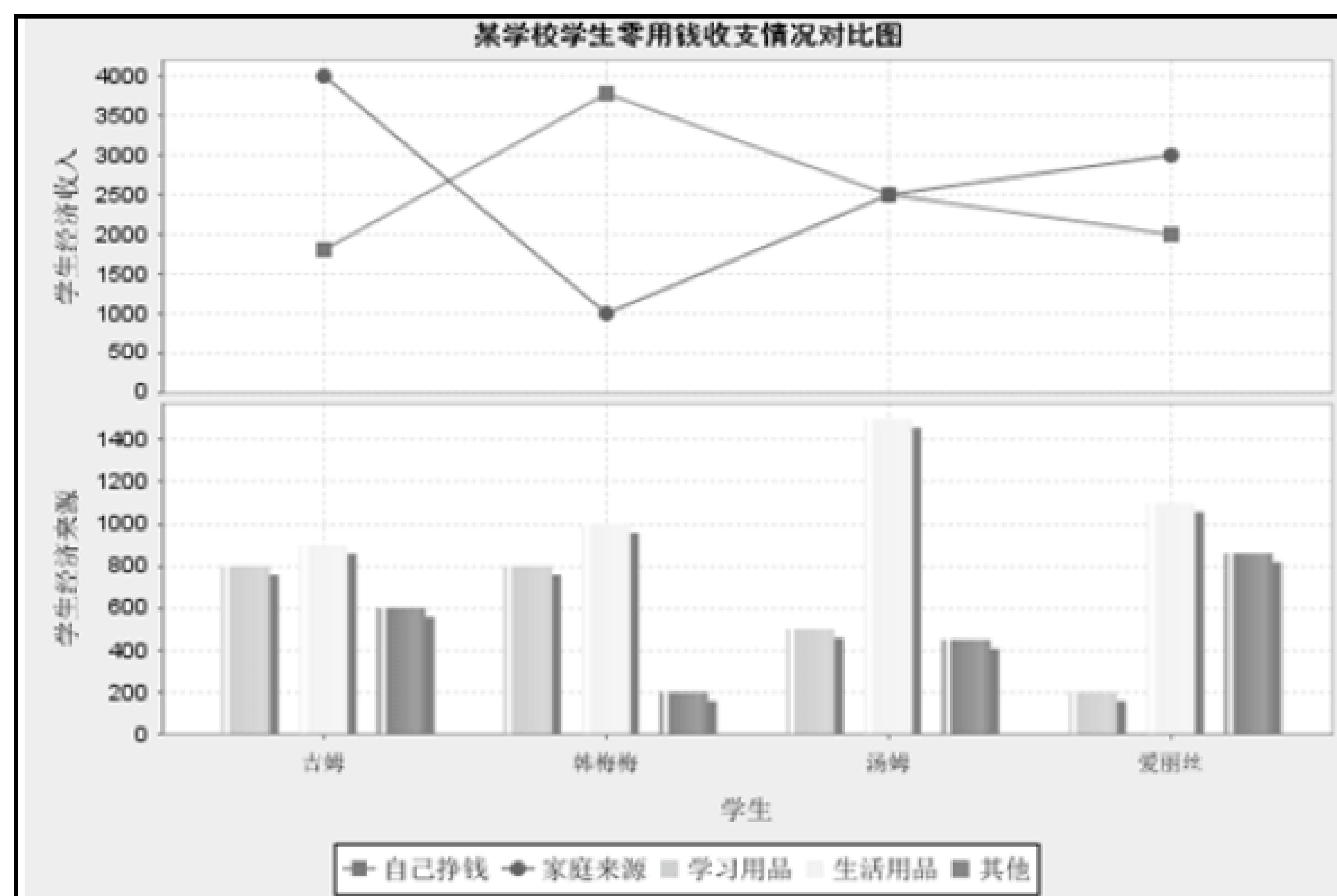


图 9.40　共用 X 轴组合图表

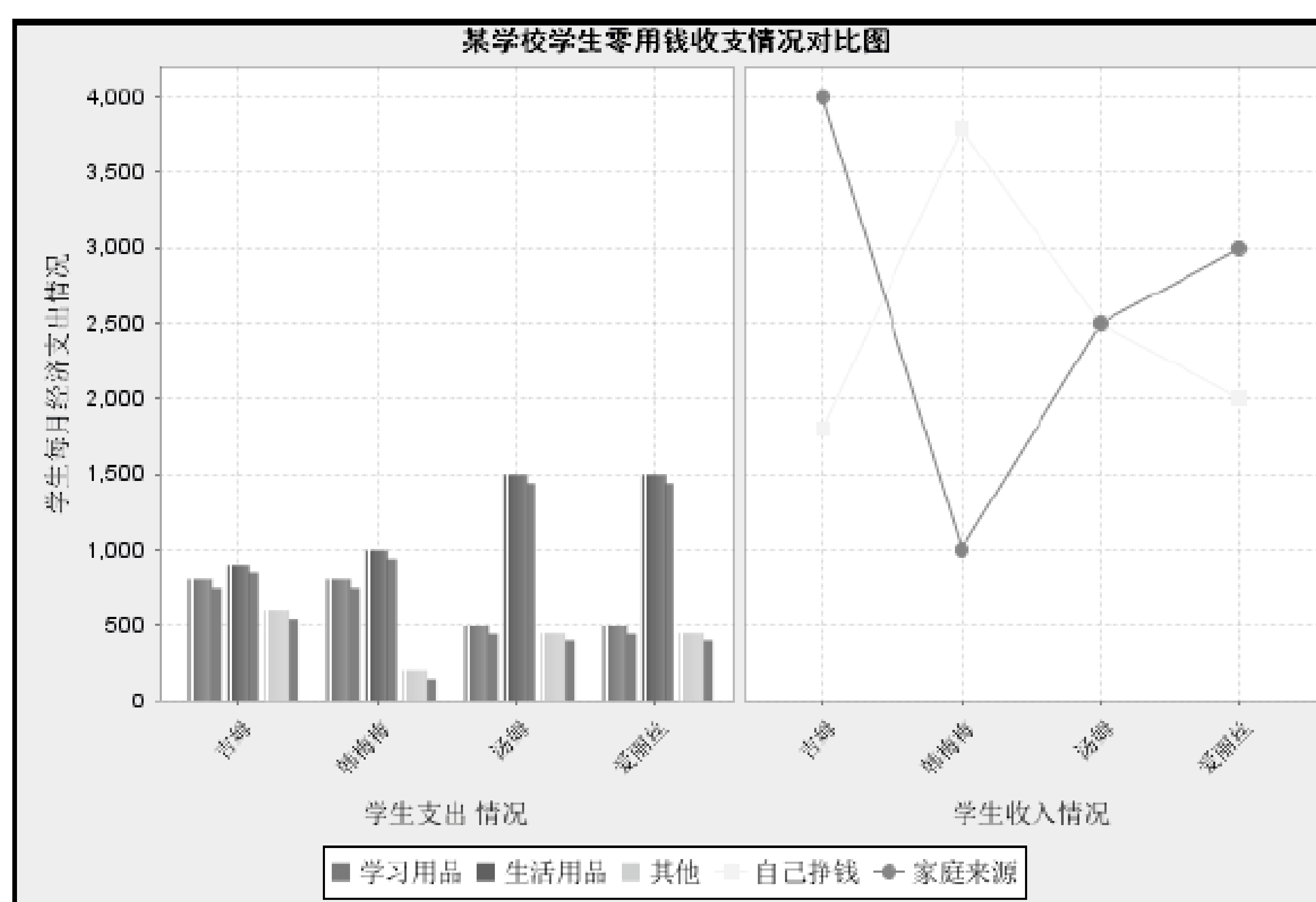


图 9.41　共用 Y 轴组合图表

关键技术

利用 JFreeChart 生成组合图时，可以应用 CombinedDomainCategoryPlot 类或 CombinedRangeCategoryPlot 类的 add()方法添加多个 Plot。CombinedDomainCategoryPlot 对象表示的是共用 X 轴图表的数据区对象，CombinedRangeCategoryPlot 对象表示的是共用 Y 轴图表的数据区对象。

设计过程

（1）创建 ChartUtil 类，编写获取图表数据源的方法。代码如下：

```
/**
 * 获得第 1 张对比图表的数据源（收入）
 * @return
 */
public static DefaultCategoryDataset getCategoryDataset_in() {
    DefaultCategoryDataset categoryDataset = new DefaultCategoryDataset();
    //自己挣钱
    categoryDataset.addValue(1800, in_s1, student1);
    categoryDataset.addValue(3800, in_s1, student2);
    categoryDataset.addValue(2500, in_s1, student3);
    categoryDataset.addValue(2000, in_s1, student4);
    //家庭来源
    categoryDataset.addValue(4000, in_s2, student1);
    categoryDataset.addValue(1000, in_s2, student2);
    categoryDataset.addValue(2500, in_s2, student3);
    categoryDataset.addValue(3000, in_s2, student4);
    return categoryDataset;
}
/**
 * 获得第 2 张对比图表的数据源（支出）
 * @return
 */
public static DefaultCategoryDataset getCategoryDataset_out() {
    DefaultCategoryDataset categoryDataset = new DefaultCategoryDataset();
    //学习用品
    categoryDataset.addValue(800, out_s1, student1);
    categoryDataset.addValue(800, out_s1, student2);
    categoryDataset.addValue(500, out_s1, student3);
    categoryDataset.addValue(200, out_s1, student4);
    //生活用品
    categoryDataset.addValue(900, out_s2, student1);
    categoryDataset.addValue(1000, out_s2, student2);
    categoryDataset.addValue(1500, out_s2, student3);
    categoryDataset.addValue(1100, out_s2, student4);
    //其他
    categoryDataset.addValue(600, out_s3, student1);
    categoryDataset.addValue(200, out_s3, student2);
```

```
        categoryDataset.addValue(450, out_s3, student3);
        categoryDataset.addValue(860, out_s3, student4);
        return categoryDataset;
}
```

（2）编写 getChart()方法，生成共用 X 轴的组合图表。代码如下：

```
public static   JFreeChart getChart() {
            //绘制第 1 张图表（收入报表）
            NumberAxis numberAxis_in = new NumberAxis("学生经济收入");
            numberAxis_in.setStandardTickUnits(NumberAxis.createIntegerTickUnits());
            LineAndShapeRenderer lineAndShapeRenderer = new LineAndShapeRenderer();
            lineAndShapeRenderer
                        .setBaseToolTipGenerator(new StandardCategoryToolTipGenerator());
            CategoryPlot categoryPlot_in = new CategoryPlot(
                        getCategoryDataset_in(), null, numberAxis_in,
                        lineAndShapeRenderer);
            categoryPlot_in.setDomainGridlinesVisible(true);
            //绘制第 2 张图表（支出报表）
            NumberAxis numberAxis_out = new NumberAxis("学生经济来源");
            numberAxis_out.setStandardTickUnits(NumberAxis.createIntegerTickUnits());
            BarRenderer barRenderer = new BarRenderer();
            barRenderer.setBaseToolTipGenerator(new StandardCategoryToolTipGenerator());
            CategoryPlot categoryPlot_out = new CategoryPlot(
                        getCategoryDataset_out(), null, numberAxis_out, barRenderer);
            categoryPlot_out.setDomainGridlinesVisible(true);
            //将两张图表合二为一
            CategoryAxis categoryAxis = new CategoryAxis("学生");
            CombinedDomainCategoryPlot combinedPlot = new CombinedDomainCategoryPlot(categoryAxis);
            combinedPlot.add(categoryPlot_in, 2);
            combinedPlot.add(categoryPlot_out, 2);
            chart = new JFreeChart("某学校学生零用钱收支情况对比图", new Font("黑体", 1, 12),
                        combinedPlot, true);
            return chart;
    }
```

（3）创建 index.jsp 页，用于显示生成的组合图表。具体代码参见配书光盘。

心法领悟 272：组合图表的作用。

组合图表可以同时展现多张图表，让用户浏览到更多的统计信息。随着各种信息海量增长，组合图表的应用越来越广。

第10章 基于Cewolf组件的图表编程

- 生成基于DefaultCategoryDataset数据集的图表
- 绘制饼状图表
- 绘制基于XYDataset数据集的图表
- 绘制基于OHLCDataset数据集的图表
- 生成组合图表
- 绘制其他类型的图表
- 综合图表的应用

10.1　生成基于 DefaultCategoryDataset 数据集的图表

实例 273　生成水平直方图

光盘位置：光盘\MR\10\273

高级

实用指数：★★★

实例说明

Cewolf 组件是开源、免费的，其实现需要有 JFreeChart 组件的支持。Cewolf 组件在 JSP 页面中通过标签来展现不同类型的图表，也就是 Cewolf 的底层依赖于 JFreeChart，然后通过 Cewolf 自定义的标签库展示图表。本实例通过 Cewolf 组件生成了水平直方图，运行结果如图 10.1 所示。

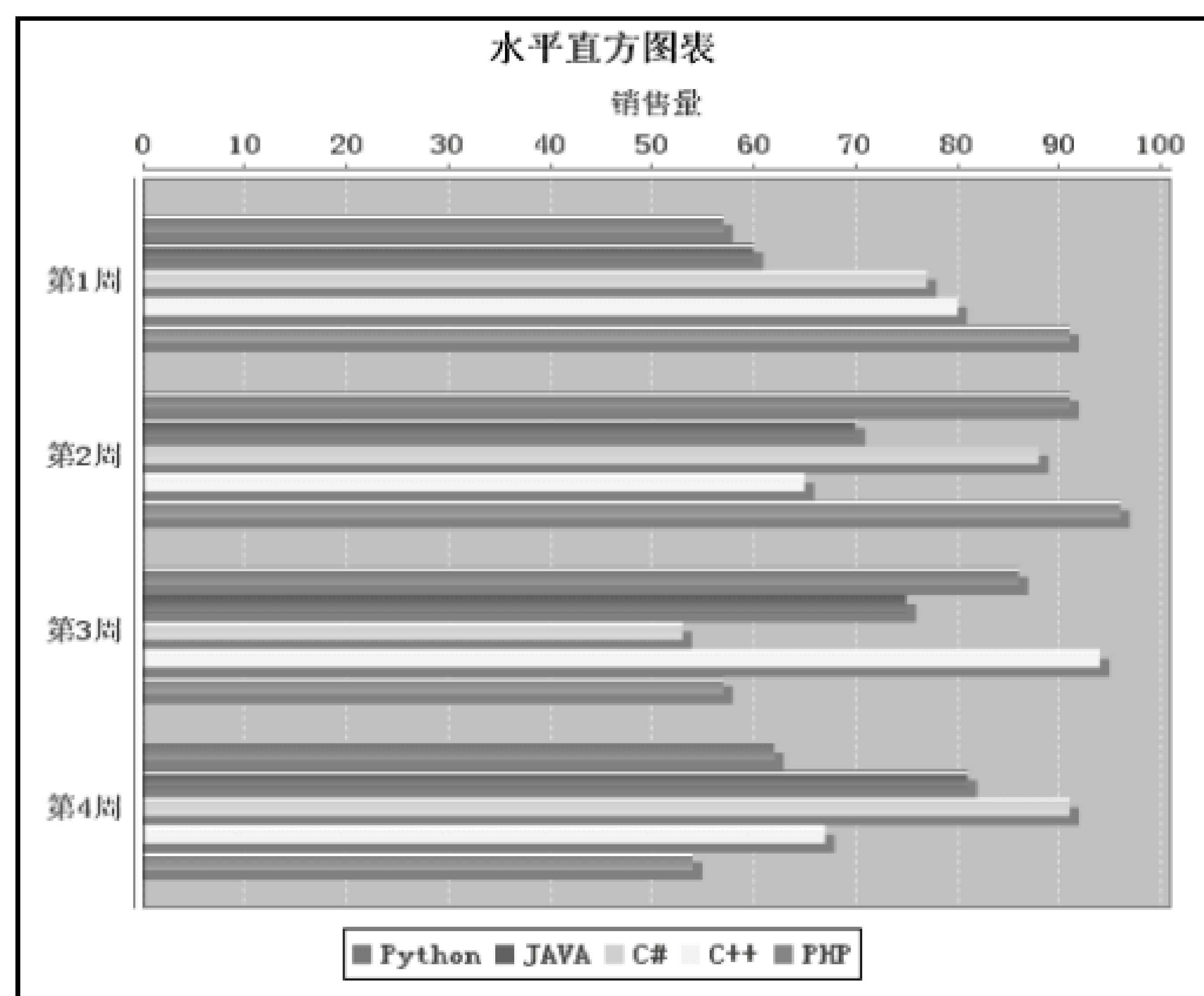


图 10.1　水平直方图

通过 Cewolf 组件生成图表，需要实现以下两个接口。

（1）de.laures.cewolf.DatasetProducer 接口

DatasetProducer 用于生成图表的数据集。在实现该接口时，需要实现以下 3 个方法。

- public Object produceDataset(Map params)：该方法用于创建并返回相应的数据集对象。例如，本实例的直方图所需的数据集对象为 DefaultCategoryDataset，然后向数据集中添加数据，最后将数据集对象返回给 Object。
- public String getProducerId()：该方法返回一个 String 字符串，表示当前数据集生产者（DatasetProducer）的标识，该标识将会被 Cewolf 标签所使用。
- public boolean hasExpired(Map params, Date since)：该方法被 Cewolf 自身所调用，用于检查当前数据集生产者（DatasetProducer）上一次生产的数据是否可以继续使用。取值为 false 时，表示不检查以前的数据；取值为 true 时，表示可以使用以前数据集生产者所创建的数据。

（2）de.laures.cewolf.ChartPostProcessor 接口

当对 Cewolf 生成的图表感觉不满意时，可以实现 ChartPostProcessor 接口，对图表进行加工处理，改变 Cewolf 生成的图表的默认显示样式。在此实现这个接口的目的主要是处理图表中文字体的乱码问题。实现 ChartPostProcessor 接口，需要实现 public void processChart (Object chart, Map params)方法。该方法就是用来处理图表的，参数 chart 就是 JFreeChart 对象，接下来的任务就是通过这个 JFreeChart 对象设置图表的样式（如标题字体、颜色、笔触等）。

设计过程

（1）想要成功应用 Cewolf 组件，需要在 web.xml 中对 Cewlof 组件提供的 Servlet 进行配置。代码如下：

```
<servlet>
	<servlet-name>CewolfServlet</servlet-name>
	<servlet-class>de.laures.cewolf.CewolfRenderer</servlet-class>
	<init-param>
		<param-name>overliburl</param-name>
		<param-value>overlib.js</param-value>
	</init-param>
	<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
	<servlet-name>CewolfServlet</servlet-name>
	<url-pattern>/cewolf/*</url-pattern>
</servlet-mapping>
```

注意： 在配置 web.xml 之前，需要将 Cewlof 提供的 overlib.js 文件复制到项目的根目录下，该文件为浏览器在浏览图像时提供了工具提示。

（2）创建 BarDatasetProducer 类并实现 DatasetProducer 接口，用于生成直方图的数据集。代码如下：

```
public class BarDatasetProducer implements de.laures.cewolf.DatasetProducer,
	Serializable {
	private String bookTitle[] = {"Python", "JAVA", "C#", "C++", "PHP"};
	private String category[] = {"第 1 周", "第 2 周", "第 3 周", "第 4 周" };
	public Object produceDataset(Map params){
		DefaultCategoryDataset dataset = new DefaultCategoryDataset();
		int bookSales;
		for (int i = 0; i < bookTitle.length; i++){
			for (int j = 0; j < category.length; j++){
				bookSales = 50+(int)(Math.random() * 50);
				dataset.addValue(bookSales, bookTitle[i], category[j]);
			}
		}
		return dataset;
	}
	public String getProducerId(){
		return "CategoryDataProducer";
	}
	public boolean hasExpired(Map params, Date since){
		return false;
	}
}
```

（3）创建 BarPostProcessor 类并实现 ChartPostProcessor 接口，在实现的 processChart()方法中设置图表的字体。代码如下：

```
public class BarPostProcessor implements ChartPostProcessor, Serializable{
	public void processChart (Object chart, Map params) {
		JFreeChart barChart = (JFreeChart) chart;
		barChart.getTitle().setFont(new Font("宋体",Font.BOLD,15));			//标题字体
		barChart.getLegend().setItemFont(new Font("宋体",Font.BOLD,12));		//分类图例字体
		CategoryPlot plot = (CategoryPlot)barChart.getPlot();
		plot.getDomainAxis().setLabelFont(new Font("宋体",Font.BOLD,12));		//X 轴标题字体
		plot.getDomainAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));	//X 轴刻度线字体
		plot.getRangeAxis().setLabelFont(new Font("宋体",Font.BOLD,12));		//Y 轴标题字体
		plot.getRangeAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));	//Y 轴刻度线字体
	}
}
```

（4）创建 index.jsp 页，用于显示图表。首先需要引用 Cewolf 组件的标签库 TLD 文件，cewolf.tld 文件存储在 Cewolf 组件 JAR 包的 META-INF 目录下。关键代码如下：

```
<%@taglib uri="http://cewolf.sourceforge.net/taglib/cewolf.tld" prefix="cewolf" %>
```

（5）在 index.jsp 中，创建 BarDatasetProducer 和 BarPostProcessor 对象，并保存在 JSP 的 pageContext 域中，然后应用 Cewolf 组件的<cewolf:chart>标签生成图表，应用<cewolf:img>标签展现图表。代码如下：

```
<%
pageContext.setAttribute("dataset",new BarDatasetProducer());			//创建数据集
```

```
pageContext.setAttribute("barPP",new BarPostProcessor());            //图表样式
%>
  <cewolf:chart type="horizontalBar"
                  id="horizontalBarChart"
                  title="水平直方图表"
                  yaxislabel="销售量">
      <cewolf:data>
            <cewolf:producer id="dataset" />
      </cewolf:data>
      <cewolf:chartpostprocessor id="barPP"></cewolf:chartpostprocessor>
  </cewolf:chart>
  <cewolf:img chartid="horizontalBarChart" height="400" width="500" renderer="/cewolf"></cewolf:img>
```

秘笈心法

心法领悟 273：实现 ChartPostProcessor 接口时的注意事项。

通过自定义的类实现该接口时，必须同时实现 Serializable 接口，因为使用 Cewolf 组件时需要用户对 ChartPostProcessor 的实现类进行序列化，否则在应用 Cewolf 组件的<cewolf:chartpostprocessor>标签调用 ChartPostProcessor 对象时将无效。



实例 274　**生成水平堆栈图**　高级

光盘位置：光盘\MR\10\274　实用指数：★★★

实例说明

堆栈图类似于直方图（柱形图），只不过它是在柱形中根据不同的颜色分块显示不同的类别。本实例将通过 Cewlof 组件生成水平堆栈图表，运行结果如图 10.2 所示。

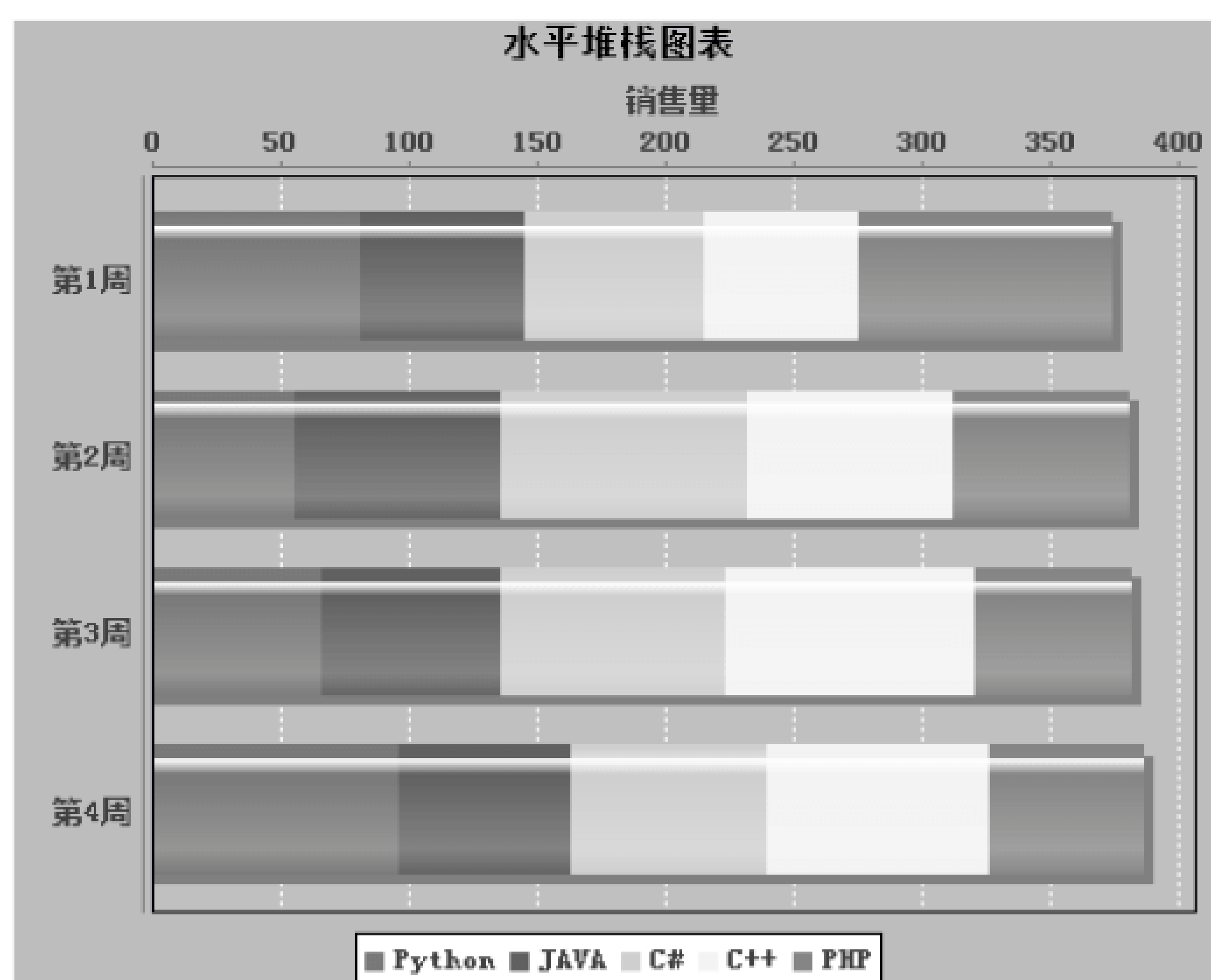


图 10.2　水平堆栈图表

堆栈图与直方图类似，都是使用 CategoryDataset 作为其数据集。因此，本实例沿用了实例 273 生成的数据集对象。接下来就是应用<cewolf:chart>标签来生成图表，用<cewolf:img>标签来展示图表。下面对这两个标签进行详细介绍。

（1）<cewolf:chart>标签

该标签用于生成所需的图表，其中包含以下几个常用属性和子标签。

- type 属性：设置生成图表的类型。Cewolf 组件能够生成的所有图表的类型可在 cewolf.tld 文件中查找。例如，饼图为 pie、3D 饼图为 pie3D、区域图为 area 等。本实例应用的是 stackedHorizontalBar 类型。

- id 属性：该属性用于为生成的图表自定义一个标识，在<cewolf:img>中需要根据这个自定义的标识查找生成的图表。
- title 属性：用于设置图表的标题。
- yaxislabel 属性：用于设置 Y 轴标题。
- xaxislabel 属性：用于设置 X 轴标题。
- antialias 属性：用于设置是否消除抗锯齿效果。取值为 false 时消除抗锯齿，默认值为 true。
- <cewolf:data>标签：用于设置图表所需的数据集来源。通过<cewolf:data>的<cewolf:producer>子标签的 id 属性设置数据集。
- <cewolf:chartpostprocessor>标签：用于处理图表效果，通过其 id 属性设置处理图表的对象来源。

（2）<cewolf:img>标签

该标签用于展示 Cewolf 组件生成的图表，其中包含以下几个常用属性。

- chartid 属性：该属性的取值必须是<cewolf:chart>标签定义的 id 值，这样才能找到要展示的图表对象。
- width 属性：用于设置图表的宽度。
- height 属性：用于设置图表的高度。
- renderer 属性：设置 Cewolf 引擎路径，<cewolf:img>需要根据该路径获取图表。该路径与 web.xml 中配置的 Cewolf 组件的 Servlet 路径相对应。

（1）创建 StackDatasetProducer 类并实现 DatasetProducer 接口，用于生成水平堆栈图的数据集。代码如下：

```
public class StackDatasetProducerimplements de.laures.cewolf.DatasetProducer,
        Serializable {
        private String bookTitle[] = {"Python", "JAVA", "C#", "C++", "PHP"};
        private String category[] = {"第 1 周", "第 2 周", "第 3 周", "第 4 周" };
        public Object produceDataset(Map params){
                DefaultCategoryDataset dataset = new DefaultCategoryDataset();
                int bookSales;
                for (int i = 0; i < bookTitle.length; i++){
                        for (int j = 0; j < category.length; j++){
                                bookSales = 50+(int)(Math.random() * 50);
                                dataset.addValue(bookSales, bookTitle[i], category[j]);
                        }
                }
                return dataset;
        }
        public String getProducerId(){
                return "CategoryDataProducer";
        }
        public boolean hasExpired(Map params, Date since){
                return false;
        }
}
```

（2）创建 StackPostProcessor 类并实现 ChartPostProcessor 接口，在实现的 processChart()方法中设置图表的字体。代码如下：

```
public class StackPostProcessor implements ChartPostProcessor, Serializable{
        public void processChart (Object chart, Map params) {
                JFreeChart barChart = (JFreeChart) chart;
                barChart.getTitle().setFont(new Font("宋体",Font.BOLD,15));                    //标题字体
                barChart.getLegend().setItemFont(new Font("宋体",Font.BOLD,12));               //分类图例字体
                CategoryPlot plot = (CategoryPlot)barChart.getPlot();
                plot.getDomainAxis().setLabelFont(new Font("宋体",Font.BOLD,12));              //X 轴标题字体
                plot.getDomainAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));          //X 轴刻度线字体
                plot.getRangeAxis().setLabelFont(new Font("宋体",Font.BOLD,12));               //Y 轴标题字体
                plot.getRangeAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));           //Y 轴刻度线字体
        }
}
```

（3）创建 index.jsp 页，用于显示图表。代码如下：

```
<%
pageContext.setAttribute("dataset",new StackDatasetProducer());      //创建数据集
pageContext.setAttribute("stackPP",new StackPostProcessor());        //加工图表
%>
  <cewolf:chart type="stackedHorizontalBar"
                            id="stackedHorizontalBar"
                            title="水平堆栈图表"
                            yaxislabel="销售量"
                            backgroundcolor="#99CC99"
                            antialias="false">
      <cewolf:data >
            <cewolf:producer id="dataset" />
      </cewolf:data>
      <cewolf:chartpostprocessor id="stackPP"></cewolf:chartpostprocessor>
  </cewolf:chart>
  <cewolf:img chartid="stackedHorizontalBar" height="400" width="500" renderer="/cewolf"></cewolf:img>
```

秘笈心法

心法领悟 274：用<cewolf:chart>标签设置图表颜色。

Cewolf 组件生成的图表背景色默认为白色，数据区背景色默认为浅灰色。利用<cewolf:chart>标签的 backgroundcolor 属性可以自定义图表背景色，利用 plotBackGroundColor 属性可以自定义 Plot 数据区的背景色。

实例 275	绘制 3D 垂直直方图 光盘位置：光盘\MR\10\275	高级 实用指数：★★★

实例说明

与普通图表相比，3D 图表更能体现出立体效果。本实例将通过 Cewlof 组件绘制 3D 垂直直方图，运行结果如图 10.3 所示。

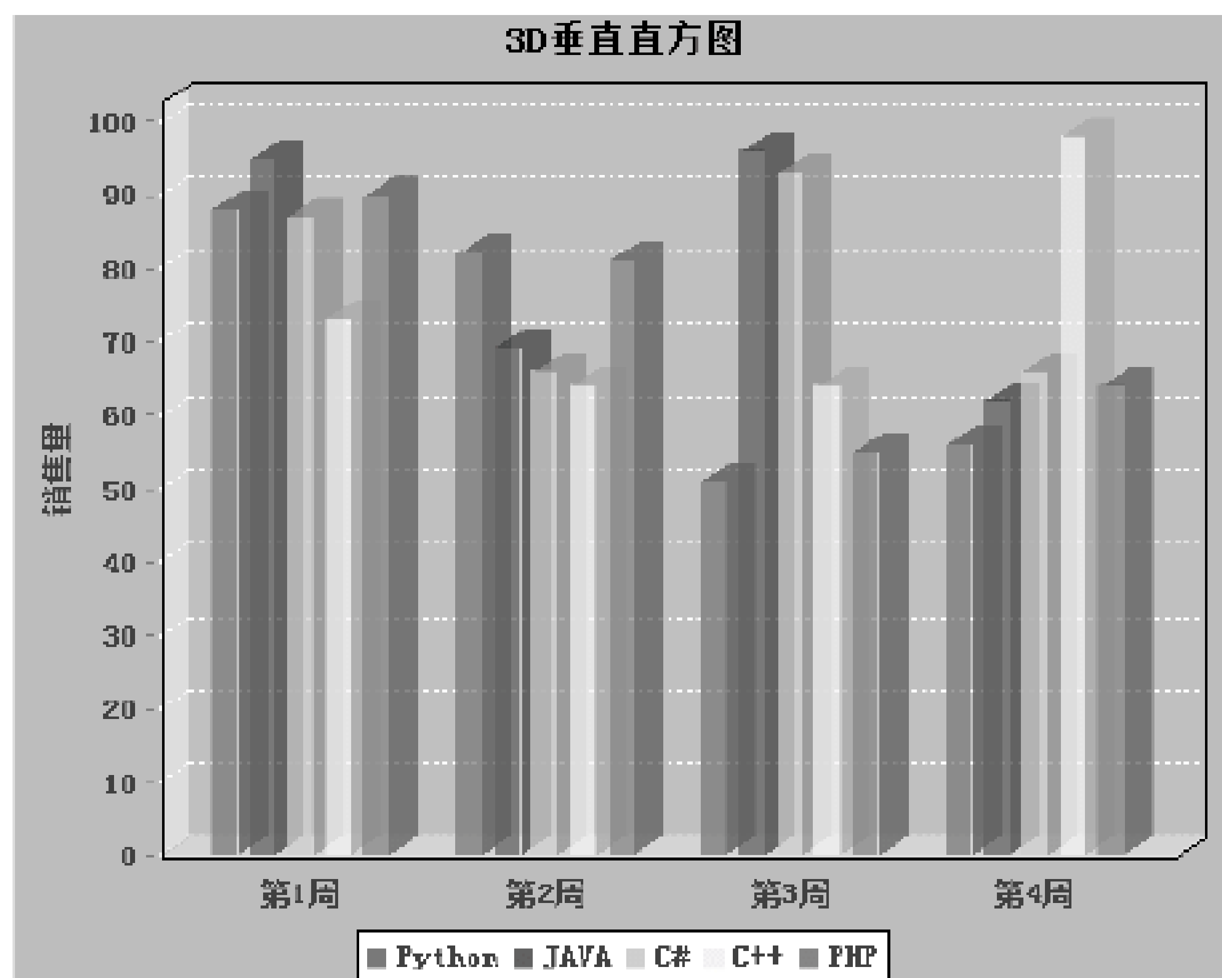


图 10.3　3D 垂直直方图表

关键技术

在 JSP 页面中，想要通过<cewolf:chart>生成 3D 垂直直方图，需要将 type 属性设置为 verticalBar3D，其他属性值与前面几个实例的设置类似。

另外，3D 垂直直方图的数据集类型同样是 CategoryDataset。

（1）创建 BarDatasetProducer 类并实现 DatasetProducer 接口，用于生成直方图的数据集。具体代码参见配书光盘。

（2）创建 BarPostProcessor 类并实现 ChartPostProcessor 接口，在实现的 processChart()方法中设置图表的字体。具体代码参见配书光盘。

（3）创建 index.jsp 页，用于显示图表。代码如下：

```
<%
pageContext.setAttribute("dataset",new BarDatasetProducer());//创建数据集
pageContext.setAttribute("3DBarPP",new BarPostProcessor());//创建数据集
%>
  <cewolf:chart type="verticalBar3D"
                        id="verticalBar3D"
                        backgroundcolor="#99CC99"
                        antialias="false"
                        title="3D 垂直直方图"
                        yaxislabel="销售量">
      <cewolf:data >
            <cewolf:producer id="dataset" />
      </cewolf:data>
      <cewolf:chartpostprocessor id="3DBarPP"></cewolf:chartpostprocessor>
  </cewolf:chart>
  <cewolf:img chartid="verticalBar3D" height="400" width="500" renderer="/cewolf"></cewolf:img>
```

秘笈心法

心法领悟 275：生成 3D 水平直方图。

将<cewolf:chart>的 type 属性设置为 verticalBar3D 可生成垂直的 3D 直方图；同样，将 type 修改为 horizontalBar3D，可以生成水平的 3D 直方图。



实例 276　生成垂直堆栈图　　高级

光盘位置：光盘\MR\10\276　　实用指数：★★★★

实例说明

前面讲过了可以通过 Cewolf 组件生成水平的堆栈图，其实通过该组件还可以生成垂直方向的堆栈图。本实例将通过 Cewolf 组件绘制垂直堆栈图，运行效果如图 10.4 所示。

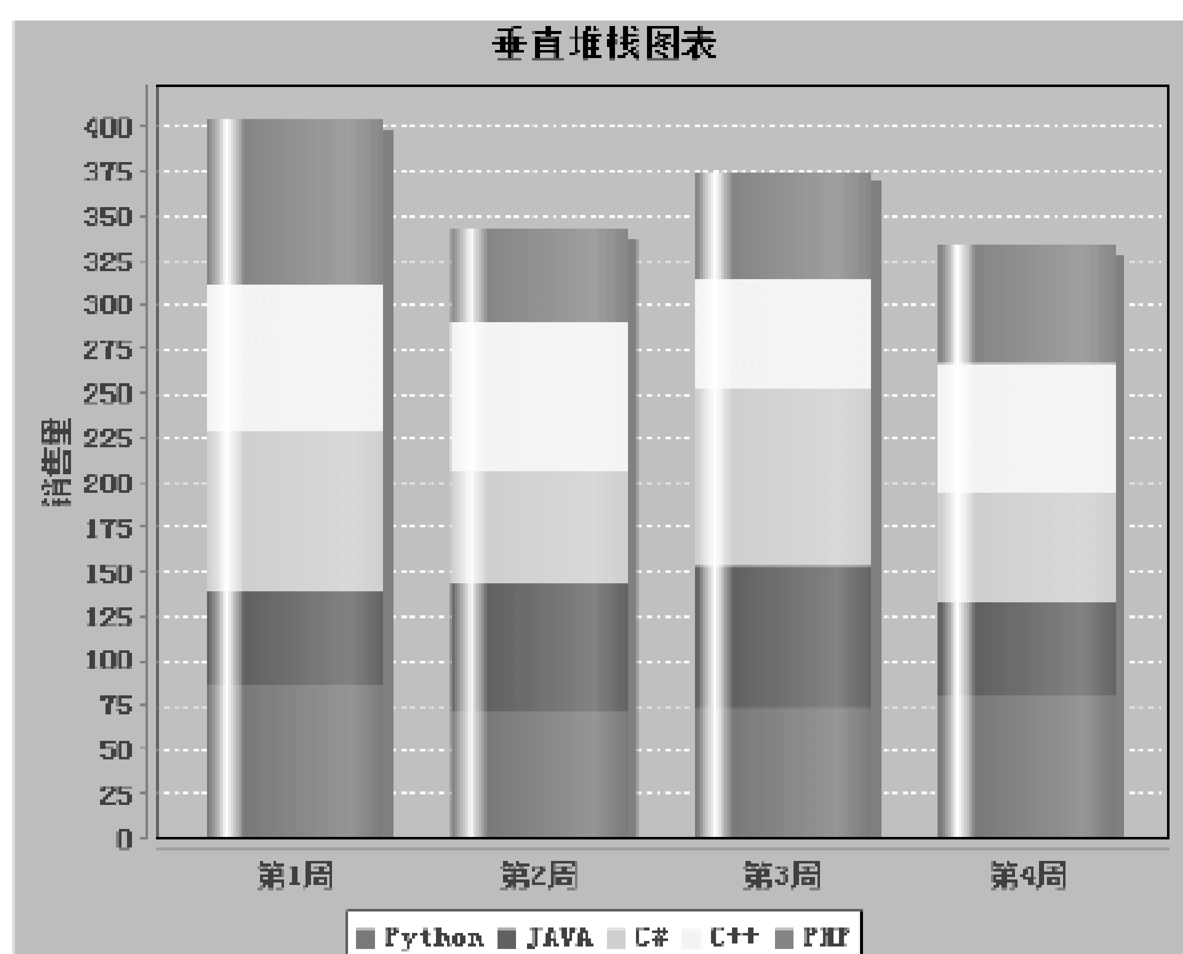


图 10.4　垂直堆栈图

在 JSP 页面中，想要通过<cewolf:chart>生成垂直堆栈图，需要将 type 属性设置为 stackedVerticalBar（与水平堆栈图表的 stackedHorizontalBar 相对应），其他属性值与前面几个实例的设置类似。

如果想生成 3D 的堆栈图，只需将<cewolf:chart>的 type 属性改为 stackedVerticalBar3D 或 stackedHorizontalBar3D 即可。

设计过程

（1）创建 StackDatasetProducer 类并实现 DatasetProducer 接口，用于生成图表的数据集。具体代码参见配书光盘。

（2）创建 StackPostProcessor 类并实现 ChartPostProcessor 接口，在实现的 processChart()方法中设置图表的字体。具体代码参见配书光盘。

（3）创建 index.jsp 页，用于显示图表。代码如下：

```
<%
pageContext.setAttribute("dataset",new StackDatasetProducer());      //创建数据集
pageContext.setAttribute("stackPP",new StackPostProcessor());        //加工图表
%>
  <cewolf:chart type="stackedVerticalBar"
                          id="stackedVerticalBar"
                          title="垂直堆栈图表"
                          yaxislabel="销售量"
                          backgroundcolor="#99CC99"
                          antialias="false" >
      <cewolf:data >
            <cewolf:producer id="dataset" />
      </cewolf:data>
      <cewolf:chartpostprocessor id="stackPP"></cewolf:chartpostprocessor>
  </cewolf:chart>
  <cewolf:img chartid="stackedVerticalBar" height="400" width="500" renderer="/cewolf"></cewolf:img>
```

秘笈心法

心法领悟 276：显示和隐藏图表的 X、Y 轴刻度值。

Cewolf 组件生成的图表默认是显示 X、Y 轴刻度值的，如果想把 X 轴的刻度值或 Y 轴的刻度值隐藏起来，可以通过<cewolf:chart>的 xticklabelsvisible 和 yticklabelsvisible 属性来设置。当 xticklabelsvisible 属性值为 false 时，将不显示 X 轴的刻度值，默认值为 true；yticklabelsvisible 属性值为 false 时，将不显示 Y 轴刻度值，默认值为 true。

实例 277	生成区域图 光盘位置：光盘\MR\10\277	高级 实用指数：★★★

本实例将通过 Cewlof 组件生成区域图（区域图的数据集类型同样为 CategoryDataset），运行结果如图 10.5 所示。

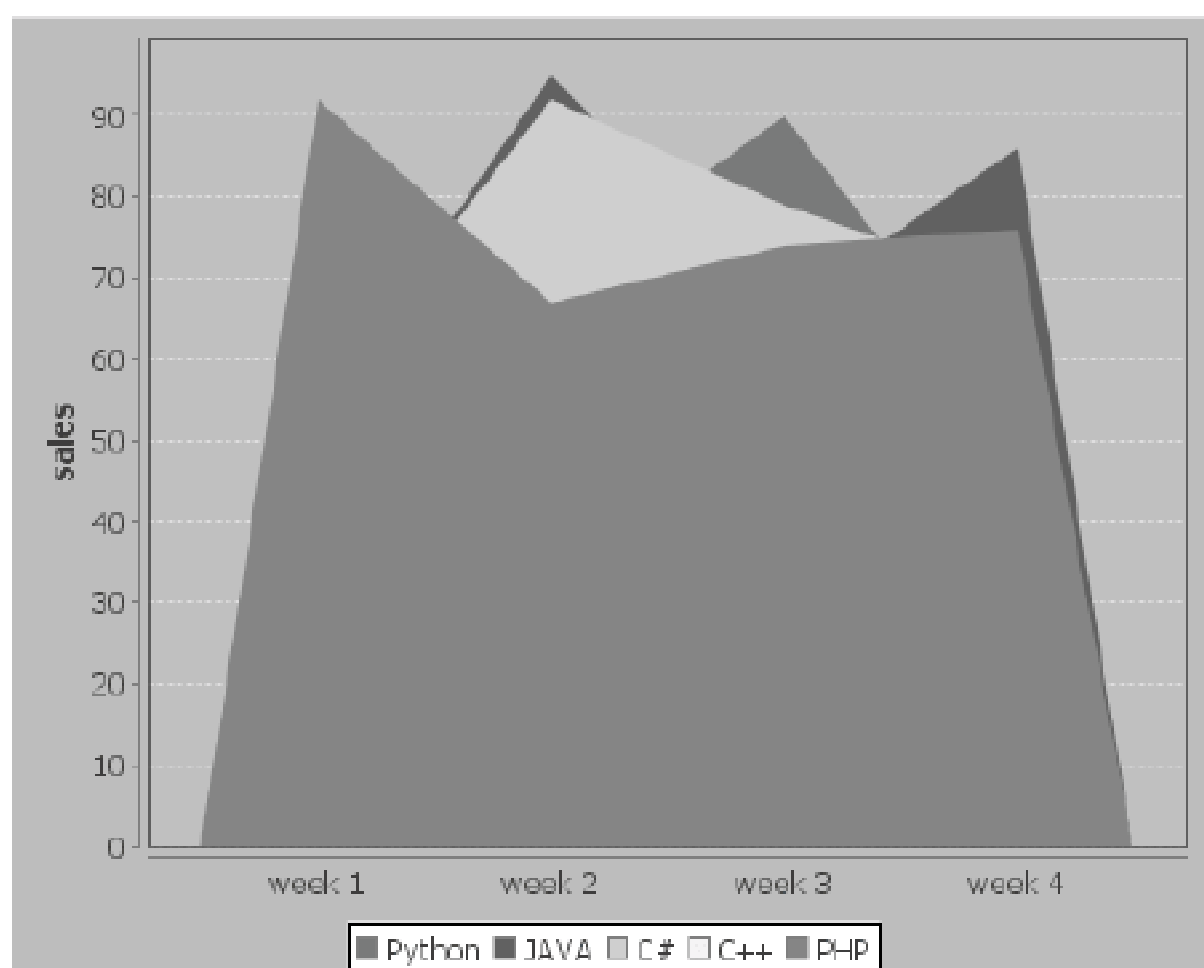


图 10.5　生成区域图

关键技术

在 JSP 页面中，想要通过<cewolf:chart>生成区域图，需要将 type 属性设置为 area，其他属性值与前面几个实例的设置类似。

设计过程

（1）创建 AreaDatasetProducer 类并实现 DatasetProducer 接口，用于生成区域图的数据集。具体代码参见配书光盘。

（2）创建 index.jsp 页，用于显示生成的图表。代码如下：

```
<%
pageContext.setAttribute("dataset",new AreaDatasetProducer());//创建数据集
%>
  <cewolf:chart type="area"
                        id="areaChart"
                        backgroundcolor="#99CC99"
                        yaxislabel="sales">
        <cewolf:data >
                <cewolf:producer id="dataset" />
        </cewolf:data>
  </cewolf:chart>
  <cewolf:img chartid="areaChart" height="400" width="500" renderer="/cewolf"></cewolf:img>
```

秘笈心法

心法领悟 277：设置图表边框。

Cewolf 组件在默认情况下生成的图表是隐藏边框的，如果希望显示边框和设置颜色，可通过<cewolf:chart>的 bordervisible 和 bordercolor 属性进行设置。bordervisible 用于设置是否显示图表的边框；bordercolor 用于设置图表的边框颜色。

10.2 绘制饼状图表

实例 278 生成普通饼图 高级

光盘位置：光盘\MR\10\278 实用指数：★★★★

与 JFreeChart 相比，应用 Cewolf 组件同样可以生成饼图，并且很方便。本实例将应用 Cewolf 组件绘制普通饼图，运行结果如图 10.6 所示。

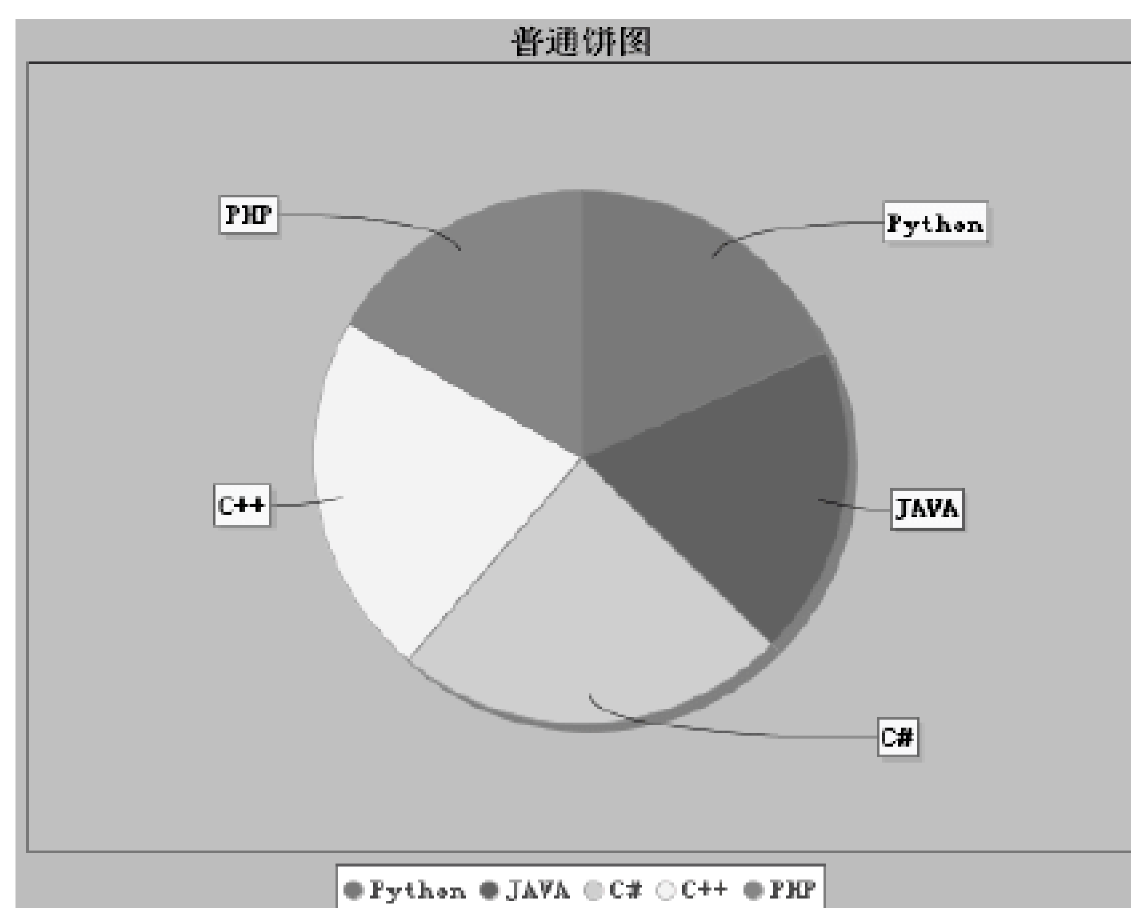


图 10.6 生成普通饼图

■ 关键技术

生成饼图时，首先应该创建饼图的数据集。饼图的数据集为 PieDataset 类型，PieDataset 是 JFreeChart 中的饼图数据集类型。只需要实现 Cewolf 组件的 DatasetProducer 接口的 produceDataset()方法，然后创建 DefaultPieDataset 对象并填充数据即可。

在应用<cewolf:chart>生成饼图时，type 属性应该设置为 pie，表示普通的饼图类型。

■ 设计过程

（1）创建 PieDatasetProducer 类并实现 DatasetProducer 接口，用于生成饼图的数据集。代码如下：

```
public class PieDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
        private    String title[] = {"Python", "JAVA", "C#", "C++", "PHP"};
        public Object produceDataset(Map params){
                DefaultPieDataset dataset = new DefaultPieDataset();                    //饼图数据集
                for (int i = 0; i < title.length; i++){
                        Random r = new Random();
                        int count = 100+(int)r.nextInt(80);
                        dataset.setValue(title[i], count);                              //添加饼图数据
                }
                return dataset;
        }
        public String getProducerId(){
                return "PieDataProducer";
        }
        public boolean hasExpired(Map params, Date since){
                return false;
        }
}
```

（2）创建 PiePostProcessor 类并实现 ChartPostProcessor 接口，用于对饼图进行处理。代码如下：

```
public class PiePostProcessor implements ChartPostProcessor, Serializable{
      public void processChart (Object chart, Map params) {
            JFreeChart pieChart = (JFreeChart) chart;
            pieChart.getTitle().setFont(new Font("宋体",Font.BOLD,15));             //标题字体
            pieChart.getLegend().setItemFont(new Font("宋体",Font.BOLD,12));      //分类图例字体
            PiePlot plot = (PiePlot)pieChart.getPlot();
            plot.setLabelFont(new Font("宋体",Font.BOLD,12));
        }
}
```

（3）创建 index.jsp 页，显示生成的饼图。代码如下：

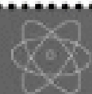
```
<%
 pageContext.setAttribute("pieData", new PieDatasetProducer());                    //创建数据集
 pageContext.setAttribute("toolTip",new ToolTip());                                //工具提示
 pageContext.setAttribute("piePP",new PiePostProcessor());                         //加工图表
%>
  <cewolf:chart type="pie"
                        id="pieChart"
                        title="普通饼图"
                        backgroundcolor="#99CC99">
        <cewolf:data >
                <cewolf:producer id="pieData" />
        </cewolf:data>
        <cewolf:chartpostprocessor id="piePP"></cewolf:chartpostprocessor>
  </cewolf:chart>
  <cewolf:img chartid="pieChart" height="400" width="500" renderer="/cewolf">
        <cewolf:map tooltipgeneratorid="toolTip"></cewolf:map>
  </cewolf:img>
```

■ 秘笈心法

心法领悟 278：饼图的工具提示。

Cewolf 组件中包含一个设置饼图工具提示的 PieToolTipGenerator 接口，如果希望饼图带有工具提示，可以实现这个接口。该接口只有一个 generateToolTip()方法，用于返回 String 类型的工具提示信息。要想显示工具提

示，还需要在<cewolf:img>标签的<cewolf:map>子标签中通过 tooltipgeneratorid 属性指定工具提示对象。



实例 279 生成 3D 饼图

光盘位置：光盘\MR\10\279

高级

实用指数：★★★★

■

与普通饼图相比，3D 饼图看起来更加直观，更具立体感。本实例将通过 Cewolf 组件生成 3D 饼图，运行结果如图 10.7 所示。

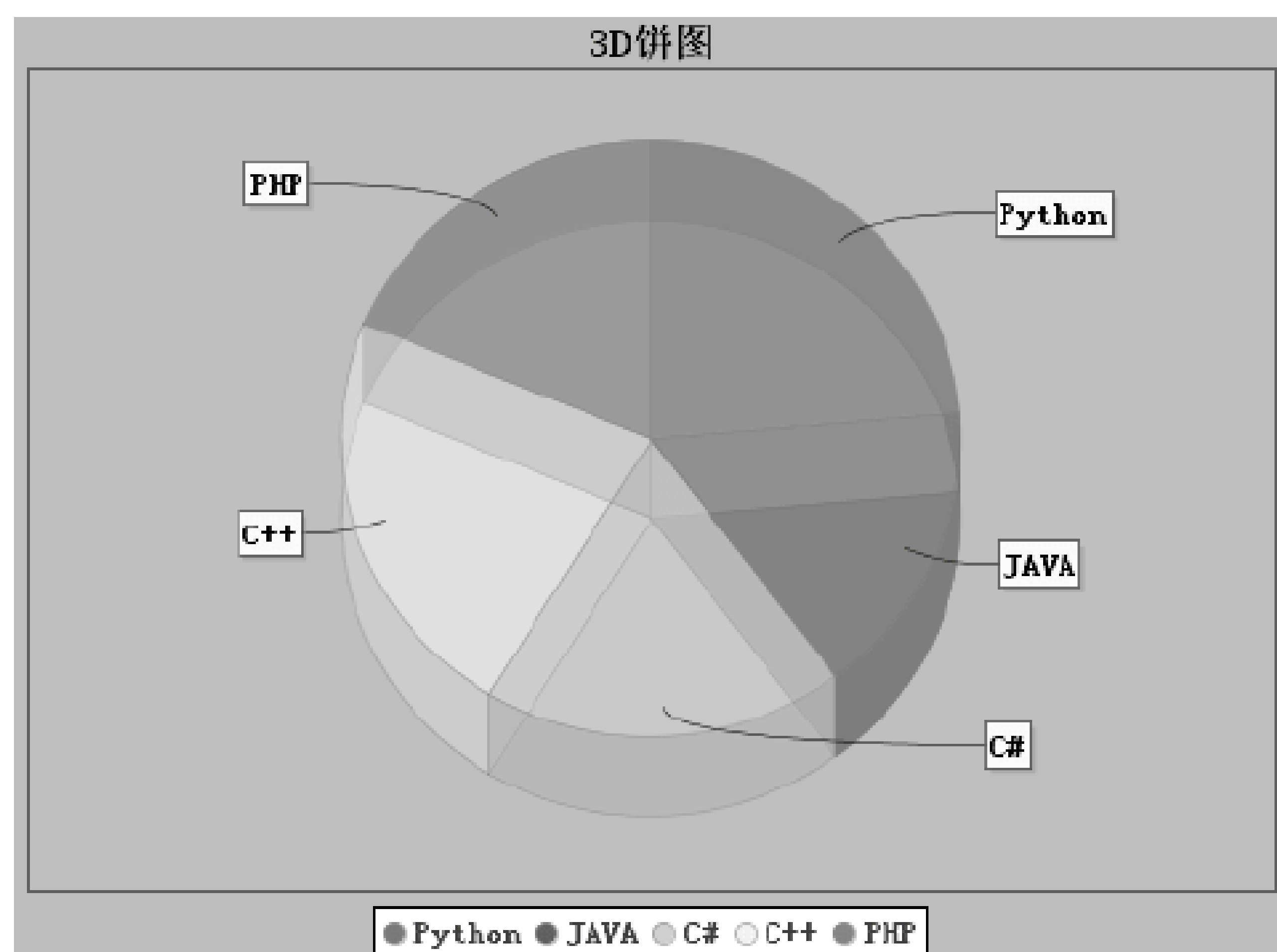


图 10.7　生成 3D 饼图

■ 关键技术

生成 3D 饼图与普通饼图基本类似，使用的都是 PieDataset 数据集；区别在于应用<cewolf:chart>生成 3D 饼图时，需要将 type 属性设置为 pie3D。

■ 设计过程

（1）创建 PieDatasetProducer 类并实现 DatasetProducer 接口，用于生成饼图的数据集。代码如下：

```
public class PieDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
    private String title[] = {"Python", "JAVA", "C#", "C++", "PHP"};
    public Object produceDataset(Map params){
        DefaultPieDataset dataset = new DefaultPieDataset();            //饼图数据集
        for (int i = 0; i < title.length; i++){
            Random r = new Random();
            int count = 100+(int)r.nextInt(80);
            dataset.setValue(title[i], count);                          //添加饼图数据
        }
        return dataset;
    }
    public String getProducerId(){
        return "PieDataProducer";
    }
    public boolean hasExpired(Map params, Date since){
        return false;
    }
}
```

（2）创建 PiePostProcessor 类并实现 ChartPostProcessor 接口，用于对饼图进行处理。代码如下：

```
public class PiePostProcessor implements ChartPostProcessor, Serializable{
    public void processChart (Object chart, Map params) {
        JFreeChart pieChart = (JFreeChart) chart;
        pieChart.getTitle().setFont(new Font("宋体",Font.BOLD,15));            //标题字体
        pieChart.getLegend().setItemFont(new Font("宋体",Font.BOLD,12));       //分类图例字体
        PiePlot    plot = (PiePlot)pieChart.getPlot();
```

```
            plot.setLabelFont(new Font("宋体",Font.BOLD,12));
            plot.setBackgroundAlpha(0.6f);              //背景透明度
            plot.setForegroundAlpha(0.4f);              //前景透明度
        }
}
```

（3）创建 index.jsp 页，显示生成的 3D 饼图。代码如下：

```
<%
pageContext.setAttribute("pieData", new PieDatasetProducer());     //创建数据集
pageContext.setAttribute("pieTip", new ToolTip());                 //工具提示
pageContext.setAttribute("piePP", new PiePostProcessor());         //加工饼图
%>
  <cewolf:chart type="pie3D"
                id="pie3DChart"
                title="3D 饼图"
                backgroundcolor="#99CC99">
      <cewolf:data >
            <cewolf:producer id="pieData" />
      </cewolf:data>
      <cewolf:chartpostprocessor id="piePP"></cewolf:chartpostprocessor>
  </cewolf:chart>
  <cewolf:img chartid="pie3DChart" height="370" width="500" renderer="/cewolf">
      <cewolf:map tooltipgeneratorid="pieTip"></cewolf:map>
  </cewolf:img>
```

秘笈心法

心法领悟 279：3D 饼图的透明度。

为了更好地体现 3D 饼图的立体效果，在程序中可以设置饼图的透明度。通过 PiePlot 对象的 setBackgroundAlpha(float alpha)方法可以设置饼图的背景透明度；通过 setForegroundAlpha(float alpha)方法可以设置饼图的前景透明度。参数 alpha 的取值范围在 0.0～1.0 之间。

10.3　绘制基于 XYDataset 数据集的图表

实例 280	生成线段图（折线图） 光盘位置：光盘\MR\10\280	高级 实用指数：★★★

线段图又称为折线图，可以清晰地显示商品数据或其他数据的行情走势。本实例将演示如何生成线段图，运行结果如图 10.8 所示。

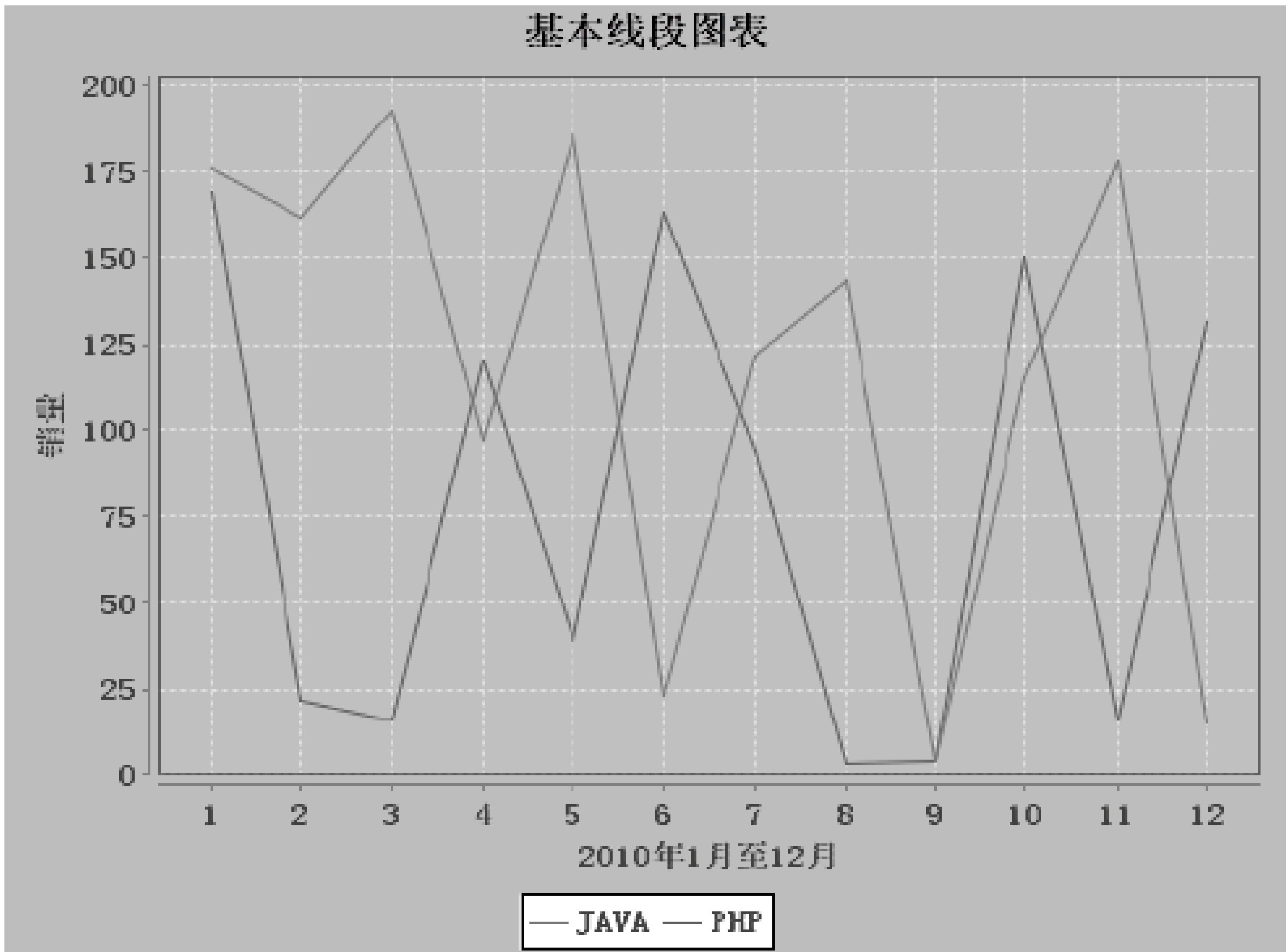


图 10.8　生成线段图表

关键技术

XY 轴线段图的数据集类型为 XYDataset，这也是 JFreeChart 中的数据集类型。数据区的 Plot 类型为 XYPlot，所以可以直接通过 JFreeChart 对象的 getXYPlot()方法返回 XY 轴的数据区 XYPlot 对象。

应用<cewolf:chart>生成普通的线段图时，应该将 type 属性设置为 xy。

（1）创建 XYLineDatasetProducer 类并实现 DatasetProducer 接口，生成 XY 线段图的数据集。代码如下：

```
public class XYLineDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
    public Object produceDataset(Map params){
        XYSeries xysJava = new XYSeries("JAVA");                    //创建数据对象
        XYSeries xysPHP = new XYSeries("PHP");                      //创建数据对象
        int javaSales = 0;
        int phpSales = 0;
        for (int month = 1; month <= 12; month++){
            javaSales =new Random().nextInt(200);
            phpSales = new Random().nextInt(200);
            xysJava.add(month, javaSales);                          //添加数据
            xysPHP.add(month, phpSales);                            //添加数据
        }
        XYSeriesCollection xysc = new XYSeriesCollection();
        xysc.addSeries(xysJava);
        xysc.addSeries(xysPHP);
        return xysc;
    }
    public String getProducerId(){
        return "XYDataProducer";
    }
    public boolean hasExpired(Map params, Date since){
        return true;
    }
}
```

（2）创建 XYLinePostProcessor 类并实现 ChartPostProcessor 接口，用于设置线段图中的字体。代码如下：

```
public class XYLinePostProcessor implements ChartPostProcessor, Serializable{
    public void processChart (Object chart, Map params) {
        JFreeChart lineChart = (JFreeChart) chart;
        lineChart.getTitle().setFont(new Font("宋体",Font.BOLD,15));                  //标题字体
        lineChart.getLegend().setItemFont(new Font("宋体",Font.BOLD,12));             //分类图例字体
        XYPlot plot = lineChart.getXYPlot();
        plot.getDomainAxis().setLabelFont(new Font("宋体",Font.BOLD,12));             //X 轴标题字体
        plot.getDomainAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));         //X 轴刻度线字体
        plot.getRangeAxis().setLabelFont(new Font("宋体",Font.BOLD,12));              //Y 轴标题字体
        plot.getRangeAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));          //Y 轴刻度线字体
}
}
```

（3）创建 index.jsp 页，显示生成的图表。代码如下：

```
<%
 pageContext.setAttribute("xyDataset", new XYLineDatasetProducer());          //创建数据集
 pageContext.setAttribute("xylinePP", new XYLinePostProcessor());             //创建数据集
%>
  <cewolf:chart type="xy"
                id="xyChart"
                xaxislabel="2010 年 1 月至 12 月"
                yaxislabel="销量"
                title="基本线段图表"
                backgroundcolor="#99CC99">
      <cewolf:data >
          <cewolf:producer id="xyDataset" />
      </cewolf:data>
      <cewolf:chartpostprocessor id="xylinePP"></cewolf:chartpostprocessor>
  </cewolf:chart>
  <cewolf:img chartid="xyChart" height="370" width="500" renderer="/cewolf">
  </cewolf:img>
```

秘笈心法

心法领悟 280：显示线段的连接点。

通过 XY 线段图的 XYPlot 对象的 getRenderer()方法可返回 XYLineAndShapeRenderer 对象，然后再通过 XYLineAndShapeRenderer 对象的 setBaseShapesVisible(boolean value)方法控制连接点的显示或隐藏。

实例 281	生成区域图 光盘位置：光盘\MR\10\281	高级 实用指数：★★★

实例说明

本实例将以 XYDataset 为数据集，生成基于 XY 轴的区域图，如图 10.9 所示。

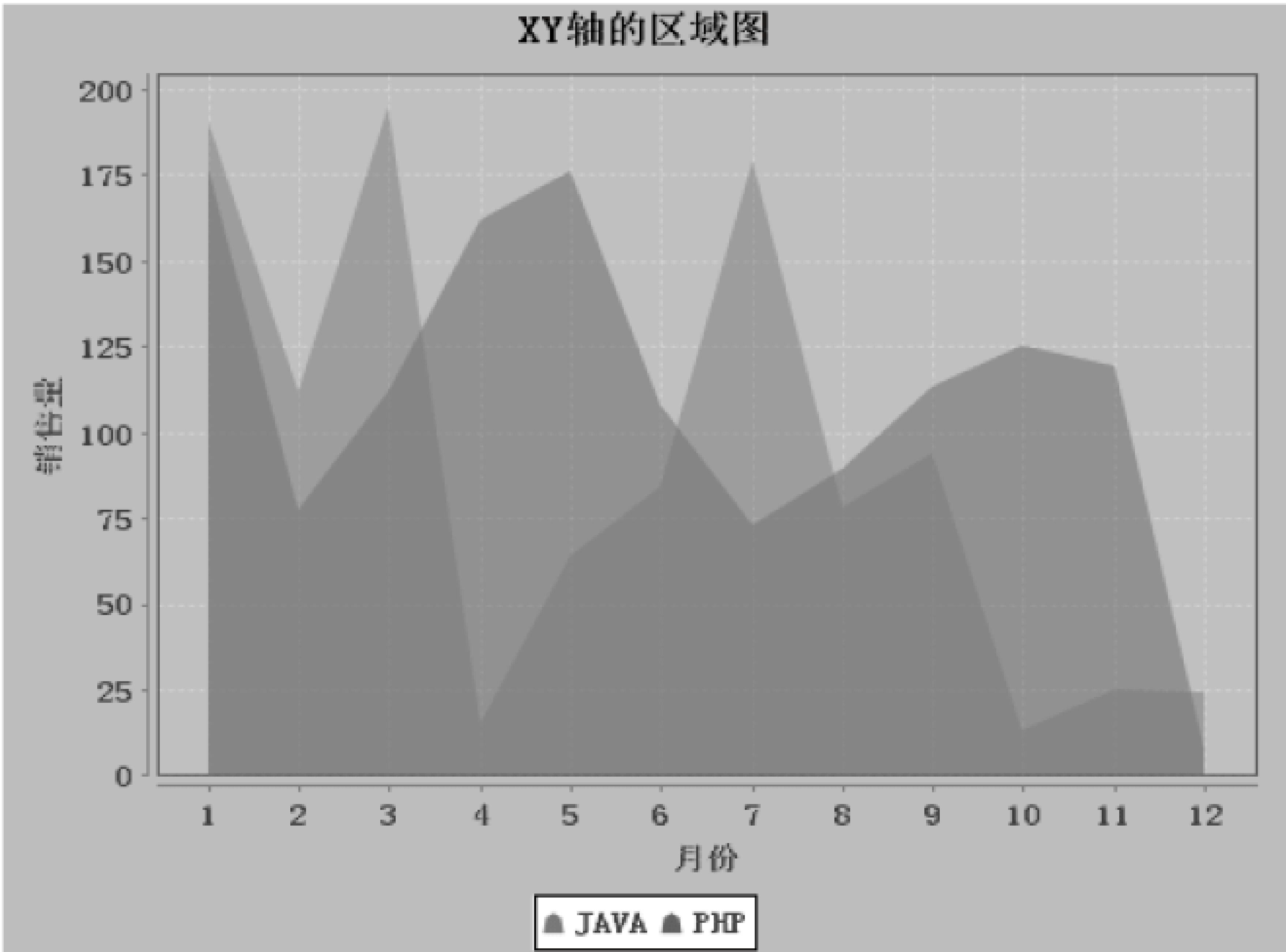


图 10.9　XY 轴的区域图

关键技术

XY 轴的区域图以 XYDataset 为数据集。在应用<cewolf:chart>生成 XY 轴区域图时，需要将 type 属性设置为 areaxy。

设计过程

（1）创建 XYLineDatasetProducer 类并实现 DatasetProducer 接口，生成区域图的数据集。具体代码参见配书光盘。

（2）创建 XYLinePostProcessor 类并实现 ChartPostProcessor 接口，用于设置图表中的字体。具体代码参见配书光盘。

（3）创建 index.jsp 页，显示生成的图表。代码如下：

```
<%
pageContext.setAttribute("xyAreaDataset", new XYLineDatasetProducer());//创建数据集
pageContext.setAttribute("xyAreaPP", new XYLinePostProcessor());
%>
  <cewolf:chart type="areaxy"
                id="xyAreaChart"
                title="XY 轴的区域图"
                xaxislabel="月份"
                yaxislabel="销售量"
```

```
                         backgroundcolor="#99CC99">
        <cewolf:data >
                <cewolf:producer id="xyAreaDataset" />
        </cewolf:data>
        <cewolf:chartpostprocessor id="xyAreaPP"></cewolf:chartpostprocessor>
</cewolf:chart>
<cewolf:img chartid="xyAreaChart" height="370" width="500" renderer="/cewolf">
</cewolf:img>
```

秘笈心法

心法领悟 281：设置图表背景颜色。

在<cewolf:chart>中，除了可以应用 backgroundcolor 属性设置背景色，还可以应用<cewolf:colorpaint>标签设置背景色，通过 color 属性设置即可。

实例 282 生成散列图

光盘位置：光盘\MR\10\282

高级

实用指数：★★★

实例说明

散列图的实质就是将线段图表去掉了线，只显示出每个连接点。本实例将通过 Cewolf 组件生成散列图，运行结果如图 10.10 所示。

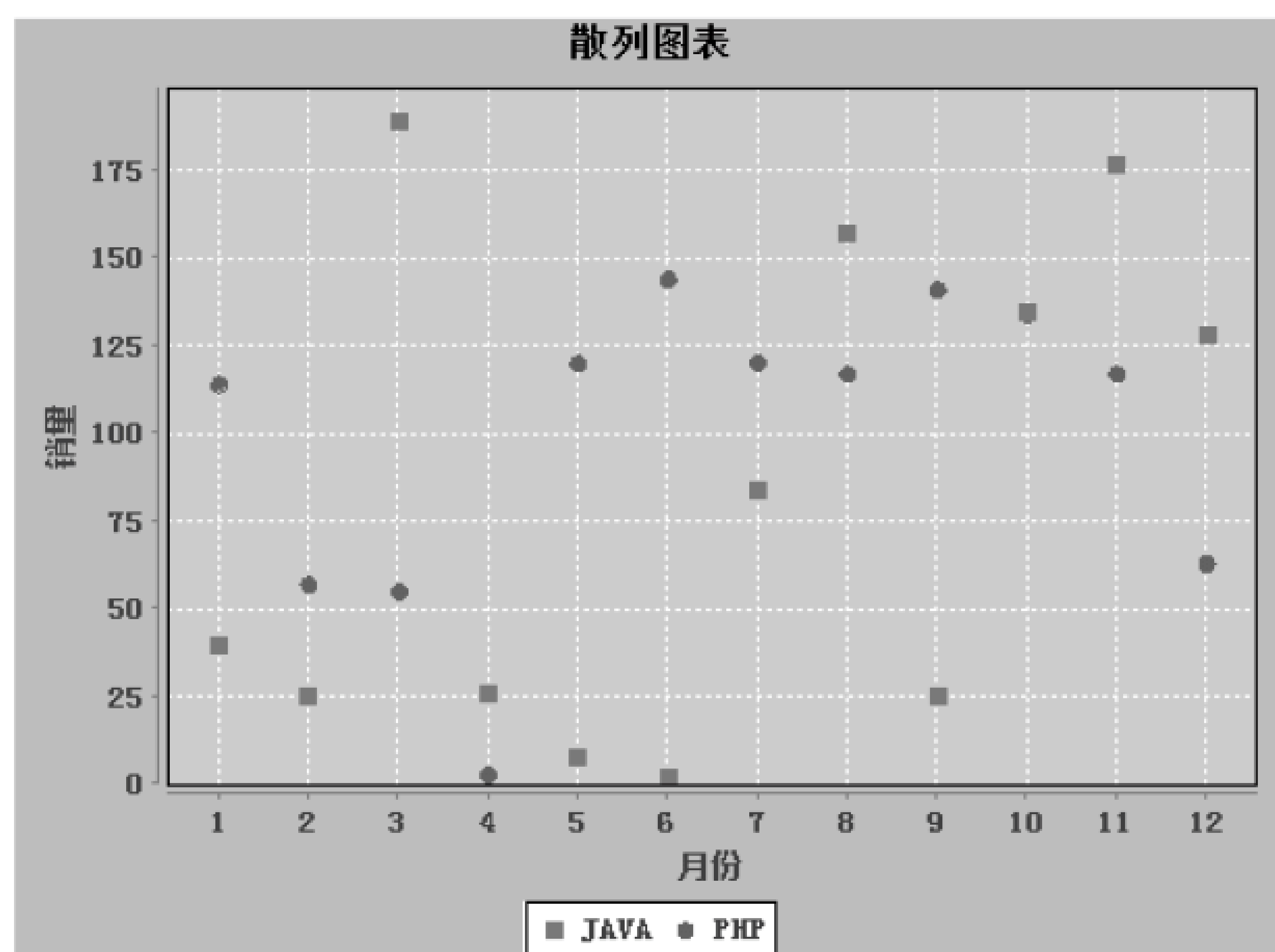


图 10.10 生成散列图

关键技术

XY 轴散列图的数据集为 XYDataset 类型，这与前两个实例是相同的。生成散列图时，需要将<cewolf:chart>的 type 属性设置为 scatter。

（1）创建 ScatterDatasetProducer 类并实现 DatasetProducer 接口，生成图表的数据集。具体代码参见配书光盘。

（2）创建 ScatterPostProcessor 类并实现 ChartPostProcessor 接口，处理图表中的字体。具体代码参见配书光盘。

（3）创建 index.jsp 页，显示生成的图表。代码如下：

```
<cewolf:chart type="scatter"
                         id="scatterChart"
                         title="散列图表"
                         xaxislabel="月份"
```

```
                    yaxislabel="销量"
                    plotbackgroundcolor="#CCCCCC"
                    backgroundcolor="#99CC99">
        <cewolf:data >
            <cewolf:producer id="xyDataset" />
        </cewolf:data>
        <cewolf:chartpostprocessor id="scatterPP"></cewolf:chartpostprocessor>
    </cewolf:chart>
    <cewolf:img chartid="scatterChart" height="370" width="500" renderer="/cewolf">
    </cewolf:img>
```

秘笈心法

心法领悟 282：设置图表的网格线颜色。

在本实例中，可以通过 XYPlot 对象的 setDomainGridlinePaint(Paint color)方法设置纵向的网格线颜色，通过 setRangeGridlinePaint(Paint color)方法设置横向的网格线颜色。

实例 283	生成时序图 光盘位置：光盘\MR\10\283	高级 实用指数：★★★

实例说明

本实例通过 Cewolf 组件生成基本的时序图，显示出 2010 年 1 月份的 Java 数据，效果如图 10.11 所示。

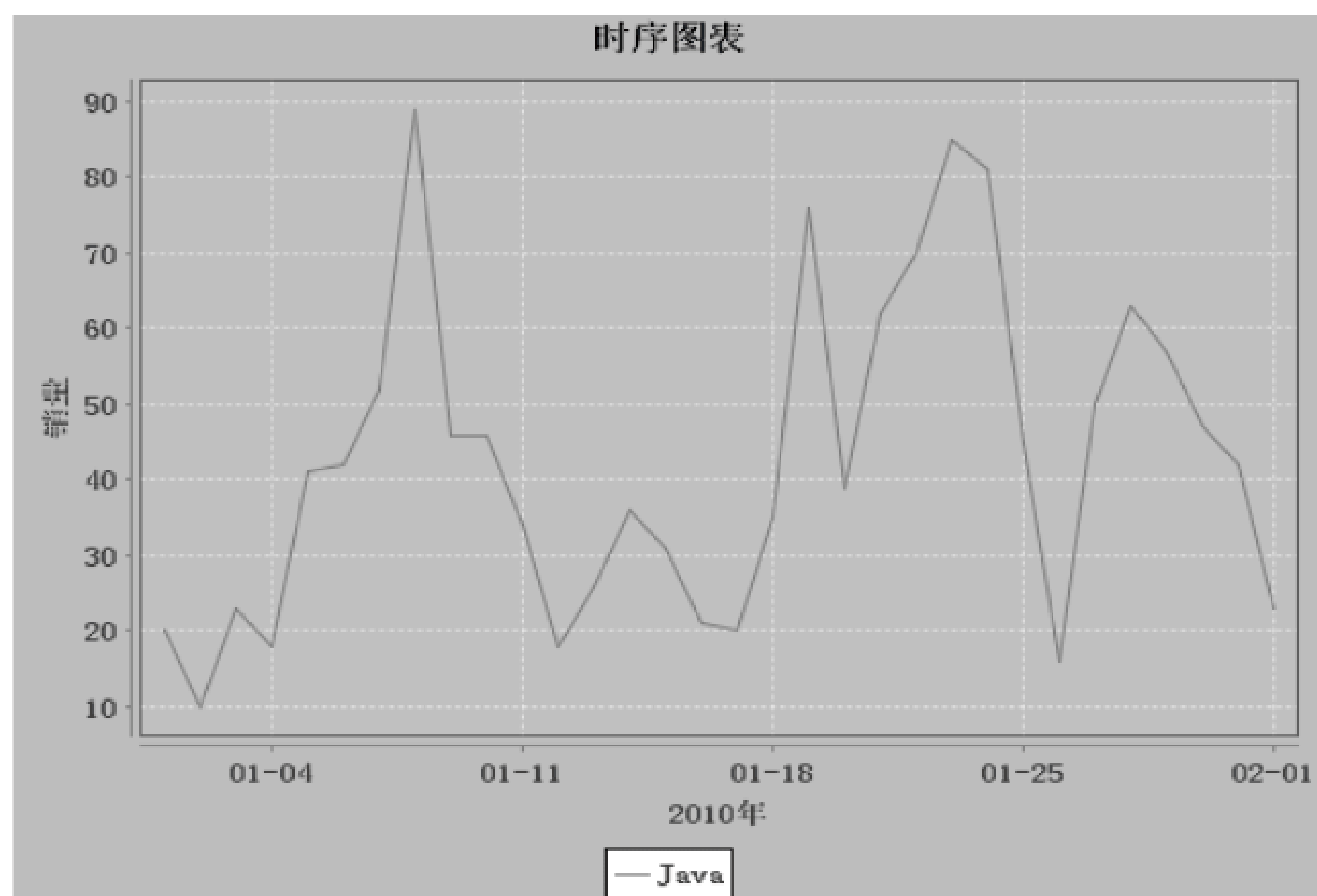


图 10.11　时序图

关键技术

时序图的数据集合类型为 TimeSeriesCollection，通过 TimeSeriesCollection 的 addSeries(TimeSeries series)方法可以向数据集中添加 TimeSeries 数据对象。向时序图中添加时序数据需要用到 TimeSeries 对象的 add()方法，该方法允许按照时间顺序添加数据。add()方法的语法如下：

```
public void add(RegularTimePeriod period,double data)
```

参数说明

❶ period：为时序图设置时间序列。RegularTimePeriod 包含几个特殊的实现类，例如，org.jfree.data.time.Day 类的对象表示特定的日期，org.jfree.data.time.Month 类的对象表示特定的月份，org.jfree.data.time.Hour 类的对象表示特定的小时，这些对象都可以作为时序图的时间序列对象。

❷ data：表示要在特定时间上添加的数据，数据类型为 double 或 number。

另外，在利用<cewolf:chart>生成时序图时，type 属性的类型为 timeseries。

设计过程

（1）创建 TimeSeriesDatasetProducer 类并实现 DatasetProducer 接口，生成时序图的数据集。代码如下：

```
public class TimeSeriesDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
        public Object produceDataset(Map params){
            TimeSeries series = new TimeSeries("Java");
             Day day = new Day(1,1,2010);
             for (int i = 0; i <= 31; i++){
                 int count = 10+new Random().nextInt(80);
                 series.add(day, count);
                 day = (Day)day.next();                              //指定下一个日期
             }
             TimeSeriesCollection xysc = new TimeSeriesCollection();
             xysc.addSeries(series);
             return xysc;
        }
        public String getProducerId(){
           return "TimeDataProducer";
        }
        public boolean hasExpired(Map params, Date since){
           return true;
        }
}
```

（2）创建 TimeSeriesPostProcessor 类并实现 ChartPostProcessor 接口，对时序图的字体和 X 轴日期进行处理。代码如下：

```
public class TimeSeriesPostProcessor implements ChartPostProcessor, Serializable{
     public void processChart (Object chart, Map params) {
          JFreeChart lineChart = (JFreeChart) chart;
          lineChart.getTitle().setFont(new Font("宋体",Font.BOLD,15));                  //标题字体
          lineChart.getLegend().setItemFont(new Font("宋体",Font.BOLD,12));             //分类图例字体
          XYPlot plot = lineChart.getXYPlot();
          plot.getDomainAxis().setLabelFont(new Font("宋体",Font.BOLD,12));             //X 轴标题字体
          plot.getDomainAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));         //X 轴刻度线字体
          plot.getRangeAxis().setLabelFont(new Font("宋体",Font.BOLD,12));              //Y 轴标题字体
          plot.getRangeAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));          //Y 轴刻度线字体
          DateAxis domainAxis = (DateAxis)plot.getDomainAxis();
          SimpleDateFormat format = new SimpleDateFormat("MM-dd");                     //日期格式器
          domainAxis.setDateFormatOverride(format);                                    //格式化日期
}
}
```

（3）创建 index.jsp 页，显示生成的图表。代码如下：

```
<cewolf:chart type="timeseries"
                  id="timeseriesChart"
                  title="时序图表"
                  xaxislabel="2010 年"
                  yaxislabel="销量"
                  backgroundcolor="#99CC99">
      <cewolf:data >
           <cewolf:producer id="timeDataset" />
      </cewolf:data>
      <cewolf:chartpostprocessor id="timeseriesPP"></cewolf:chartpostprocessor>
  </cewolf:chart>
  <cewolf:img chartid="timeseriesChart" height="370" width="550" renderer="/cewolf">
  </cewolf:img>
```

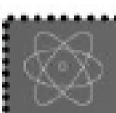
心法领悟 283：设置时间显示格式。

使用 DateAxis 类的 setDateFormatOverride()方法可以格式化时间轴显示格式。语法如下：

```
setDateFormatOverride(DateFormat formatter)
```

参数说明

formatter：表示时序图的显示格式。



实例 284　生成直方图　高级

光盘位置：光盘\MR\10\284　实用指数：★★★

实例说明

此处将要介绍的直方图表与前面实例中所介绍的直方图有所不同，这里要介绍的是 XY 轴的直方图，所使用的数据集与时序图的数据集是相同的。本实例将介绍如何通过 Cewolf 生成直方图，运行结果如图 10.12 所示。

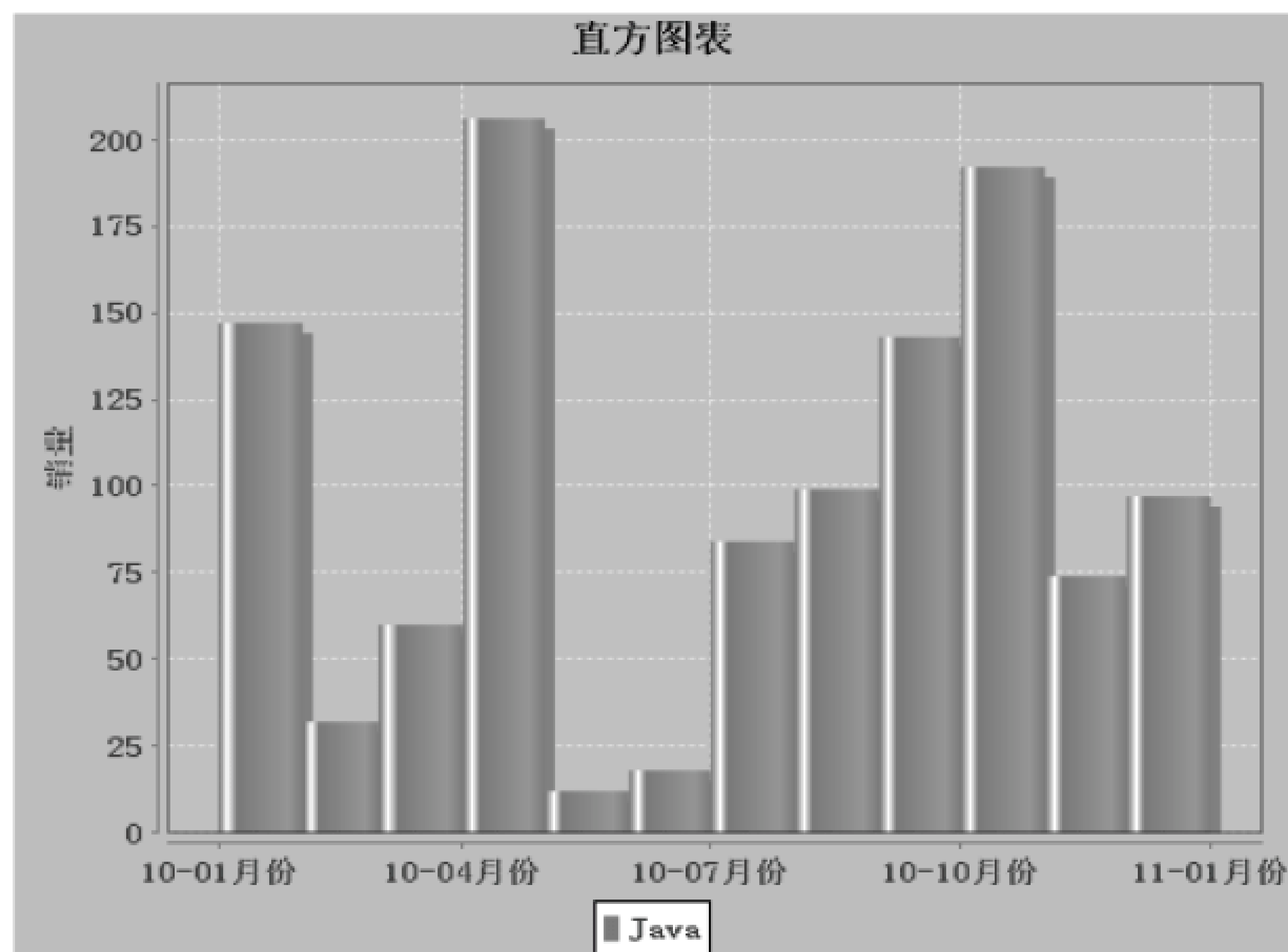


图 10.12　直方图表

关键技术

本实例实现的直方图与时序图类似，都是分析一段时间内数据的变化情况，所以使用的数据集也就是时序图的数据集。在应用<cewolf:chart>生成直方图时，需要将 type 属性设置为 verticalXYBar。

（1）创建 TimeSeriesDatasetProducer 类并实现 DatasetProducer 接口，生成图表的数据集。具体代码参见配书光盘。

（2）创建 TimeSeriesPostProcessor 类并实现 ChartPostProcessor 接口，对图表的字体和 X 轴日期进行处理。具体代码参见配书光盘。

（3）创建 index.jsp 页，显示生成的图表。代码如下：

```
<%
pageContext.setAttribute("timeDataset", new TimeseriesDatasetProducer());        //创建数据集
pageContext.setAttribute("timeseriesPP", new TimeseriesPostProcessor());          //处理图表
%>
  <cewolf:chart type="verticalXYBar"
                    id="verticalXYChart"
                    title="直方图表"
                    yaxislabel="销量"
                    backgroundcolor="#99CC99">
      <cewolf:data >
            <cewolf:producer id="timeDataset" />
      </cewolf:data>
      <cewolf:chartpostprocessor id="timeseriesPP"></cewolf:chartpostprocessor>
  </cewolf:chart>
  <cewolf:img chartid="verticalXYChart" height="370" width="500" renderer="/cewolf">
  </cewolf:img>
```

秘笈心法

心法领悟 284：设置直方图的颜色。

使用 XYItemRenderer 类的 setSeriesPaint()方法可以设置直方图指定系列的颜色。语法如下：

```
public void setSeriesPaint(int series, Paint paint)
```

参数说明

❶ series：表示指定直方图。

❷ paint：表示指定区域图的颜色。

10.4 绘制基于 OHLCDataset 数据集的图表

实例 285 生成 K 线图 光盘位置：光盘\MR\10\285 高级 实用指数：★★

实例说明

K 线图是一种常用于分析金融数据的图表，其数据集是 OHLCDataset 类型。本实例将演示如何绘制 K 线图，运行结果如果 10.13 所示。

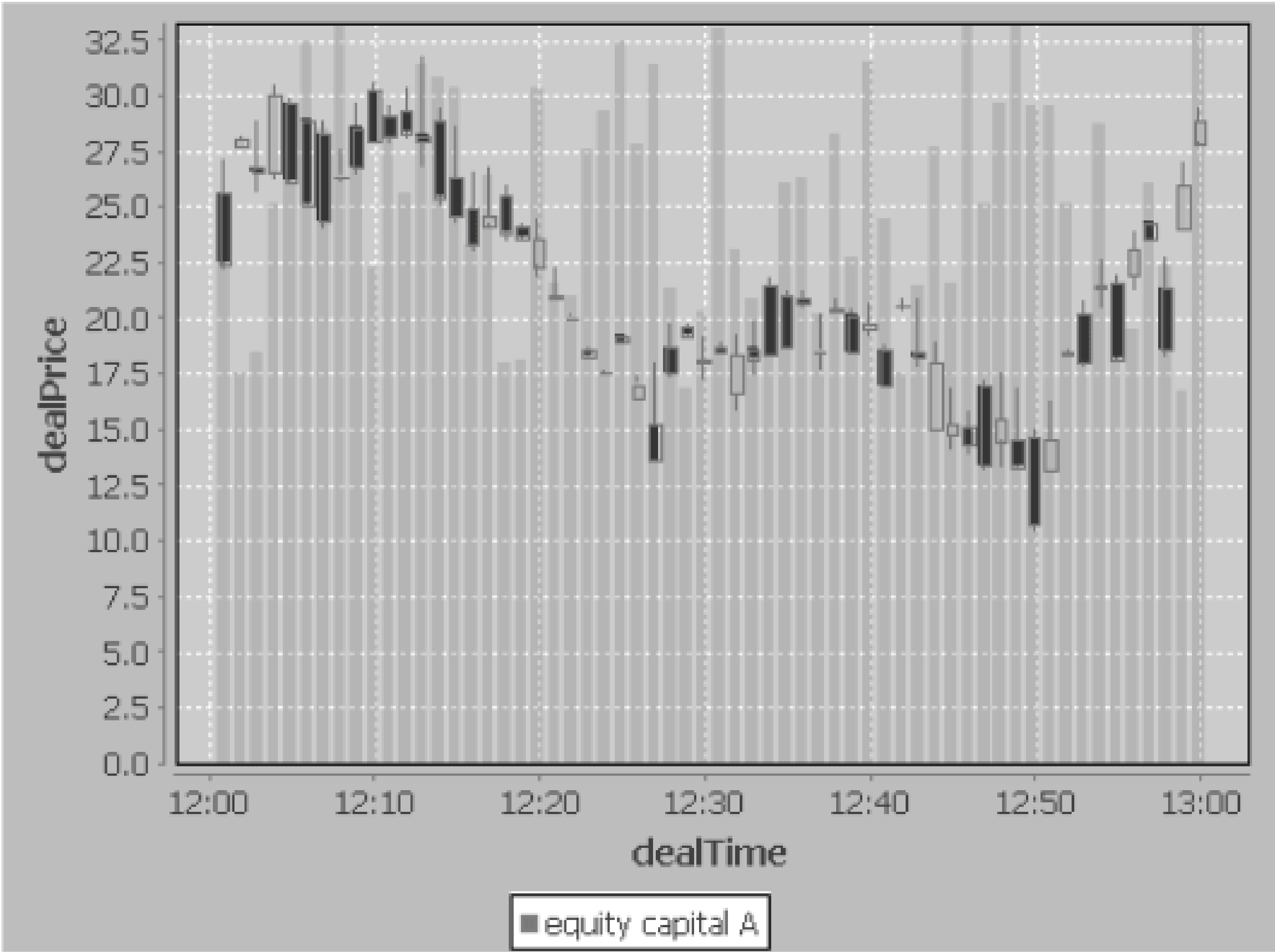


图 10.13 生成 K 线图

关键技术

K 线图中的每个数据点一般需要 6 个数据，即日期（Date）、开盘价（Open）、最高价（High）、最低价（Low）、收盘价（Close）和成交量（Volume）。JFreeChart 专门提供了一个 OHLCDataset 接口来表示这种数据结构。可以创建 DefaultHighLowDataset 对象作为 K 线图的数据集，DefaultHighLowDataset 是 OHLCDataset 接口的实现类。构造方法如下：

```
public DefaultHighLowDataset(Comparable seriesKey,Date[] date,double[] high,double[] low,double[] open,double[] close,double[] volume)
```

参数说明

❶ seriesKey：数据集名称。

❷ date：为每个数据点设置一个日期时间。

❸ high：设置一组表示最高价的数据。

❹ low：设置一组表示最低价的数据。

❺ open：设置一组开盘价数据。

❻ close：设置一组收盘价数据。

❼ volume：设置成交量。

设计过程

（1）创建 CandleStickDatasetProducer 类并实现 DatasetProducer 接口，生成 K 线图的数据集。代码如下：

```
public class CandleStickDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
 public Object produceDataset(Map params){
       int itemCount = 60;
       Date[]dates = new Date[itemCount];
       double[]high = new double[itemCount];
       double[]low = new double[itemCount];
       double[]open = new double[itemCount];
       double[]close = new double[itemCount];
       double[]volume = new double[itemCount];
       Calendar cal = new GregorianCalendar();
       //初始化第一组数据
       high[0] = 27.0;
       low[0] = 24.0;
       open[0] = 24.0;
       close[0] = 26.0;
       dates[0] = cal.getTime();
       volume[0] = 500.0 + Math.random() * 500;
       for (int i = 1; i < itemCount; i++){                                    //循环生成数据集的随机数据
          cal.roll(Calendar.MINUTE, 1);
          dates[i] = cal.getTime();
          high[i] = high[i - 1] + Math.random() * 5-2.5;
          low[i] = high[i] - (Math.random() * 5);
          open[i] = high[i] - (Math.random() * (high[i] - low[i]));
          close[i] = low[i] + (Math.random() * (high[i] - open[i]));
          volume[i] = 500.0 + Math.random() * 500;
       }
       OHLCDataset ds = new DefaultHighLowDataset(
          "equity capital A", dates, high, low, open,
          close, volume);
       return ds;
   }
   public String getProducerId(){
       return "HiLoDataProducer";
   }
   public boolean hasExpired(Map params, Date since){
       return false;
   }
}
```

（2）创建 index.jsp 页，显示生成的图表。代码如下：

```
<%
 pageContext.setAttribute("dataset", new CandleStickDatasetProducer());          //创建数据集
%>
   <cewolf:chart type="candleStick"
                        id="candleStickChart"
                        xaxislabel="dealTime"
                        yaxislabel="dealPrice"
                        backgroundcolor="#99CC99">
        <cewolf:data >
                <cewolf:producer id="dataset" />
        </cewolf:data>
   </cewolf:chart>
   <cewolf:img chartid="candleStickChart" height="370" width="500" renderer="/cewolf">
   </cewolf:img>
```


心法领悟 285：JFreeChart 的 K 线图。

Cewolf 组件是依赖于 JFreeChart 的，表示 K 线图数据集的 OHLCDataset 接口也是 JFreeChart 中的，JFreeChart 生成 K 线图时需要应用 ChartFactory 工厂的 createCandlestickChart()方法。



实例 286 生成高低图（HighLow）

光盘位置：光盘\MR\10\286

高级

实用指数：★★

实例说明

高低图与 K 线图类似，都是应用 OHLCDataset 作为数据集，因此可以直接用 K 线图的数据集作为高低图的数据集。运行效果如图 10.14 所示。

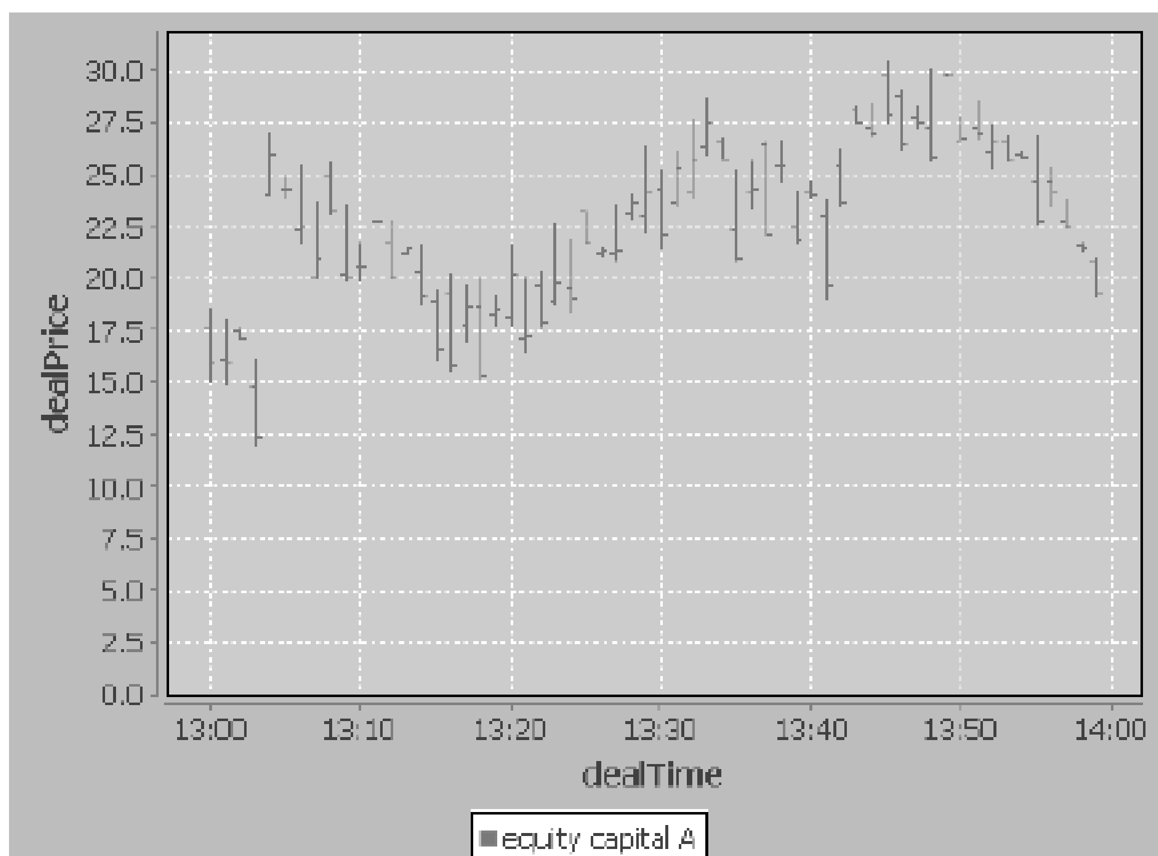


图 10.14　生成高低图

关键技术

利用 Cewolf 的<cewolf:chart>标签生成高低图时，需要将 type 属性设置为 highLow，其他的与 K 线图没有区别。

（1）创建 HighLowDatasetProducer 类并实现 DatasetProducer 接口，生成高低图的数据集。代码如下：

```
public class HighLowDatasetProducerimplements de.laures.cewolf.DatasetProducer,Serializable {
 public Object produceDataset(Map params){
        int itemCount = 60;
        Date[]dates = new Date[itemCount];
        double[]high = new double[itemCount];
        double[]low = new double[itemCount];
        double[]open = new double[itemCount];
        double[]close = new double[itemCount];
        double[]volume = new double[itemCount];
        Calendar cal = new GregorianCalendar();
        //初始化第一组数据
        high[0] = 27.0;
        low[0] = 24.0;
        open[0] = 24.0;
        close[0] = 26.0;
        dates[0] = cal.getTime();
        volume[0] = 500.0 + Math.random() * 500;
        for (int i = 1; i < itemCount; i++){    //循环生成数据集的随机数据
           cal.roll(Calendar.MINUTE, 1);
           dates[i] = cal.getTime();
           high[i] = high[i - 1] + Math.random() * 5-2.5;
           low[i] = high[i] - (Math.random() * 5);
           open[i] = high[i] - (Math.random() * (high[i] - low[i]));
           close[i] = low[i] + (Math.random() * (high[i] - open[i]));
           volume[i] = 500.0 + Math.random() * 500;
        }
        OHLCDataset ds = new DefaultHighLowDataset(
           "equity capital A", dates, high, low, open,
           close, volume);
        return ds;
```

```
    }
    public String getProducerId(){
        return "HiLoDataProducer";
    }
    public boolean hasExpired(Map params, Date since){
        return false;
    }
}
```

（2）创建 index.jsp 页，显示生成的图表。代码如下：

```
<cewolf:chart type="highLow"
                    id="highLowChart"
                    antialias="false"
                    xaxislabel="dealTime"
                    yaxislabel="dealPrice"
                    backgroundcolor="#99CC99">
    <cewolf:data >
            <cewolf:producer id="dataset" />
    </cewolf:data>
</cewolf:chart>
<cewolf:img chartid="highLowChart" height="370" width="500" renderer="/cewolf">
</cewolf:img>
```

秘笈心法

心法领悟 286：设置数据集的时间。

本实例在生成数据集时，创建的 Date 类的对象表示的是分钟。这里创建的是一个 dates 数组，该数组中的对象是通过 Calendar 类的对象的 getTime()方法获取的。Calendar 对象的 roll()方法表示每次向后滚动 1 分钟，因为指定其第一个参数为表示分钟的 Calendar 类的静态变量 Calendar.MINUTE；第二个参数表示第一个参数对象的滚动幅度，该参数为 1，表示向后滚动 1 分钟，如果该参数为负数，则向前滚动相应的间隔。

10.5　生成组合图表

实例 287　生成水平组合图表　　高级

光盘位置：光盘\MR\10\287　　实用指数：★★★

利用 Cewolf 同样可以生成组合图表，本实例将通过 Cewolf 实现时序图和直方图的组合图表，运行结果如图 10.15 所示。

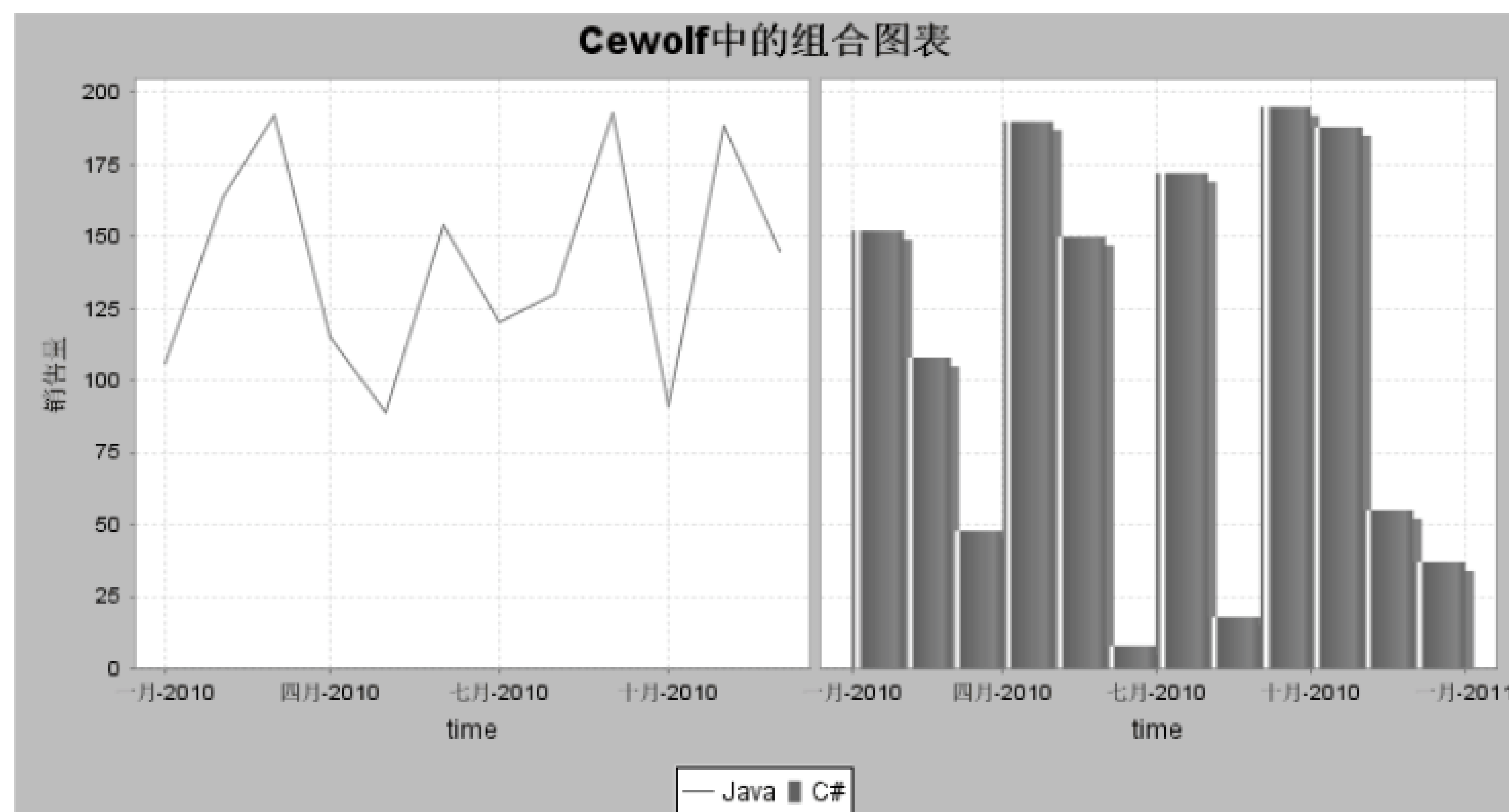


图 10.15　水平组合图表

关键技术

生成组合图表，应用的是 Cewolf 的<cewolf:combinedchart>标签。用户需要根据 type 属性指定图表的绘制风格为 combinedxy。

除了将<cewolf:combinedchart>标签的 type 属性指定为 combinedxy，还需要通过<cewolf:plot>标签为组合图表指定多个 Plot（<cewolf:plot>标签有一个 type 属性用于设置 Plot 的类别），然后通过<cewolf:data>标签设置 Plot 所需的数据集。

（1）创建 XYLineDatasetProducer 类并实现 DatasetProducer 接口，为组合图表中的时序图创建数据集。代码如下：

```
public class XYLineDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
  public Object produceDataset(Map params){
          TimeSeries series1 = new TimeSeries("Java",Month.class);
            for (int i = 1; i <= 12; i++){
              int count1 = new Random().nextInt(200);
              series1.add(new Month(i,2010), count1);
            }
            TimeSeriesCollection xysc = new TimeSeriesCollection();
            xysc.addSeries(series1);
            return xysc;
   }
   public String getProducerId(){
      return "TimeDataProducer";
   }
   public boolean hasExpired(Map params, Date since){
      return true;
   }
}
```

（2）创建 XYBarDatasetProducer 类并实现 DatasetProducer 接口，为组合图表中的直方图创建数据集。代码如下：

```
public class XYBarDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
  public Object produceDataset(Map params){
          TimeSeries series = new TimeSeries("C#",Month.class);
            for (int i = 1; i <= 12; i++){
              int count = new Random().nextInt(200);
              series.add(new Month(i,2010), count);
            }
            TimeSeriesCollection xysc = new TimeSeriesCollection();
            xysc.addSeries(series);
            return xysc;
   }
   public String getProducerId(){
      return "TimeDataProducer";
   }
   public boolean hasExpired(Map params, Date since) {
      return true;
   }
}
```

（3）创建 index.jsp 页，显示生成的组合图表。代码如下：

```
<%
 pageContext.setAttribute("barDataset", new XYBarDatasetProducer());
 pageContext.setAttribute("lineDataset", new XYLineDatasetProducer());
%>
<cewolf:combinedchart
        id="combinedChart"
        layout="horizontal"
        type="combinedxy"
        title="Cewolf 中的组合图表"
        yaxislabel="销售量">
    <cewolf:colorpaint color="#99CC99"/>
```

```
        <cewolf:plot type="xyline" xaxislabel="time" >
            <cewolf:data>
                <cewolf:producer id="lineDataset" />
            </cewolf:data>
        </cewolf:plot>
        <cewolf:plot type="xyverticalbar" xaxislabel="time">
            <cewolf:data>
                <cewolf:producer id="barDataset" />
            </cewolf:data>
        </cewolf:plot>
</cewolf:combinedchart>
<cewolf:img chartid="combinedChart" renderer="/cewolf" width="700" height="375"/>
```

秘笈心法

心法领悟 287：设置图表网格线的颜色和笔触。

通过 XYPlot 中的 setDomainGridlinePaint()和 setRangeGridlinePaint()方法可以设置网格线的颜色，通过 setDomainGridlineStroke()和 setRangeGridlineStroke()方法可以设置网格线的笔触。



实例 288　**生成垂直组合图表**　高级

光盘位置：光盘\MR\10\288　实用指数：★★★

实例说明

本实例将通过 Cewolf 组件生成垂直组合图表（包括一个时序图和一个直方图），运行结果如图 10.16 所示。

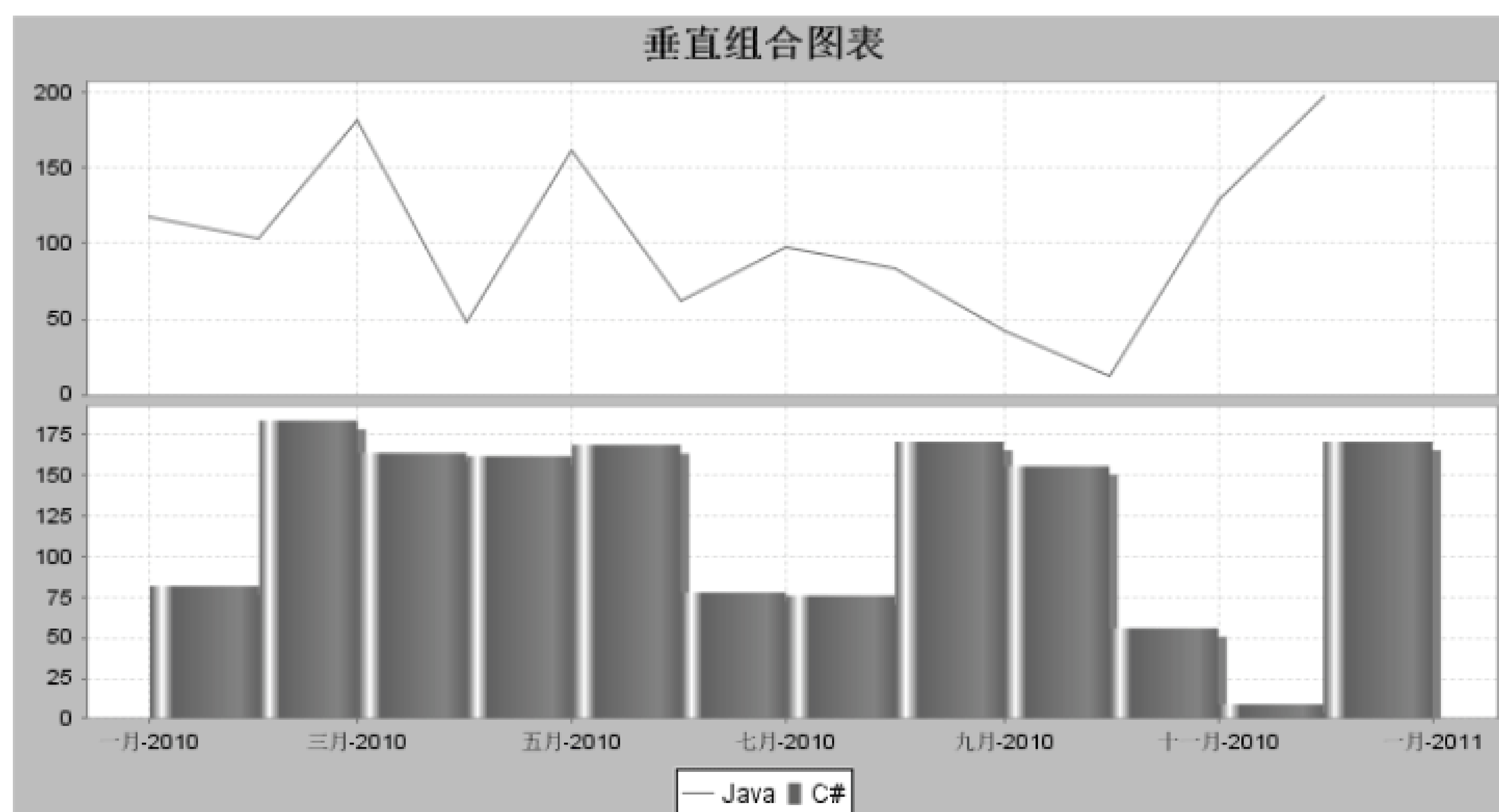


图 10.16　垂直组合图表

关键技术

垂直方向的组合图表与水平方向的组合图表的唯一区别就是图表的显示方向。生成组合图表的<cewolf:combinedchart>标签包含一个 layout 属性，该属性有两个取值，分别是 vertical（垂直方向）和 horizontal（水平方向）。

设计过程

（1）创建 XYLineDatasetProducer 类并实现 DatasetProducer 接口，为组合图表中的时序图创建数据集。

（2）创建 XYBarDatasetProducer 类并实现 DatasetProducer 接口，为组合图表中的直方图创建数据集。

（3）创建 index.jsp 页，显示生成的垂直组合图表。代码如下：

```
<%
pageContext.setAttribute("barDataset", new XYBarDatasetProducer());
pageContext.setAttribute("lineDataset", new XYLineDatasetProducer());
```

```
%>
<cewolf:combinedchart
        id="combinedChart"
        layout="horizontal"
        type="combinedxy"
        title="Cewolf 中的组合图表"
        yaxislabel="销售量">
    <cewolf:colorpaint color="#99CC99" />
        <cewolf:plot type="xyline" xaxislabel="time" >
                <cewolf:data>
                        <cewolf:producer id="lineDataset" />
                </cewolf:data>
        </cewolf:plot>
        <cewolf:plot type="xyverticalbar" xaxislabel="time">
                <cewolf:data>
                        <cewolf:producer id="barDataset" />
                </cewolf:data>
        </cewolf:plot>
</cewolf:combinedchart>
<cewolf:img chartid="combinedChart" renderer="/cewolf" width="700" height="375"/>
```

秘笈心法

心法领悟 288：组合图表的字体。

Cewolf 的组合图表是基于 X、Y 轴的，所以可以创建 XYPlot 对象的相关方法来设置图表 X、Y 轴标题或刻度的字体样式。

10.6 绘制其他类型的图表

实例 289 生成甘特图 高级

光盘位置：光盘\MR\10\289 实用指数：★★★

实例说明

甘特图表主要用来体现工作流程、软件的开发进度等。本实例以员工的工作流程为例，介绍如何通过 Cewolf 组件生成甘特图，运行结果如图 10.17 所示。

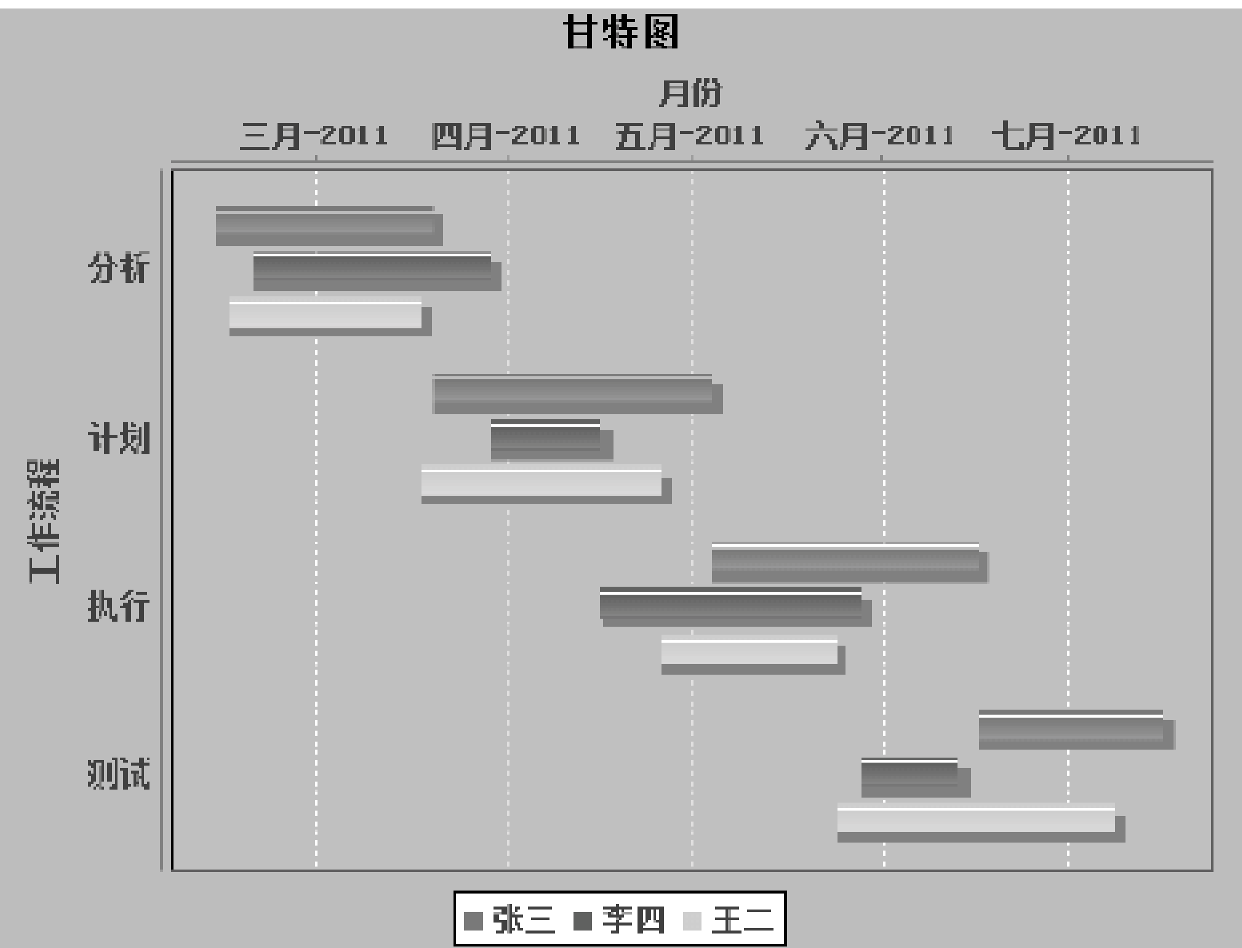


图 10.17 生成甘特图表

绘制甘特图时所需数据集是一个比较特殊的数据集，为 IntervalCategoryDataset 接口类型。由于 TashSeriesCollection 类实现了 IntervalCategoryDataset 接口，因此利用 TashSeriesCollection 表示甘特图的数据集

即可。

创建 TashSeriesCollection 类的对象后，需要创建 TashSeries 类的对象作为其数据对象，一个 TashSeries 类的对象表示一个任务计划。然后通过 TashSeriesCollection 对象的 add(TashSeries series)方法添加 TashSeries 的对象即可。

另外，在应用<cewolf:chart>生成甘特图时，应该将 type 属性设置为 gantt。

（1）创建 GanttDatasetProducer 类并实现 DatasetProducer 接口，用于创建甘特图的数据集。代码如下：

```
public class GanttDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
        final private long now = Calendar.getInstance().getTimeInMillis();
        final private long day = 1000 * 60 * 60 * 24;
        final private String[] workflows ={"分析", "计划", "执行", "测试"};
        final private String[] person ={"张三", "李四", "王二"};
        public Object produceDataset(Map params){
              TaskSeriesCollection ds = new TaskSeriesCollection();
              for (int j = 0; j < 3; j++){
                 TaskSeries ser = new TaskSeries(person[j]);
                 long lastEnd = now + getRandomTime();
                 for (int i = 0; i < workflows.length; i++){
                    long myEnd = lastEnd + getRandomTime();
                    Task t = new Task(workflows[i], new SimpleTimePeriod(new Date(lastEnd), new Date(myEnd)));
                    ser.add(t);
                    lastEnd = myEnd;
                 }
                 ds.add(ser);
              }
              return ds;
   }
   private long getRandomTime() {
           return day * (long)(Math.random() * 31+15);
   }
   public String getProducerId(){
           return "GanttDataProducer";
   }
   public boolean hasExpired(Map params, Date since) {
           return true;
   }
}
```

（2）创建 GanttPostProcessor 类并实现 ChartPostProcessor 接口，用于处理图表的字体。代码如下：

```
public class GanttPostProcessor implements ChartPostProcessor, Serializable{
      public void processChart (Object chart, Map params) {
              JFreeChart ganttChart = (JFreeChart) chart;
              ganttChart.getTitle().setFont(new Font("宋体",Font.BOLD,15));              //标题字体
              ganttChart.getLegend().setItemFont(new Font("宋体",Font.BOLD,12));     //分类图例字体
              CategoryPlot plot = (CategoryPlot)ganttChart.getPlot();
              plot.getDomainAxis().setLabelFont(new Font("宋体",Font.BOLD,12));
              plot.getDomainAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));
              plot.getRangeAxis().setLabelFont(new Font("宋体",Font.BOLD,12));
              plot.getRangeAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));
}
}
```

（3）创建 index.jsp 页，显示生成的甘特图。代码如下：

```
<%
pageContext.setAttribute("dataset",new GanttDatasetProducer());
pageContext.setAttribute("ganttPP", new GanttPostProcessor());
%>
 <cewolf:chart
        id="gantt"
        type="gantt"
        xaxislabel="工作流程"
        antialias="false"
        title="甘特图"
        yaxislabel="月份">
```

```
    <cewolf:colorpaint color="#99CC99"/>
    <cewolf:data>
        <cewolf:producer id="dataset" />
    </cewolf:data>
    <cewolf:chartpostprocessor id="ganttPP"></cewolf:chartpostprocessor>
</cewolf:chart>
<cewolf:img  chartid="gantt" renderer="/cewolf" width="500" height="375"></cewolf:img>
```

秘笈心法

心法领悟 289：设置甘特图的分类图形颜色。

首先通过 CategoryPlot 的 getRenderer()方法获取 CategoryItemRenderer 类的对象，然后通过 CategoryItemRenderer 类的对象的 setSeriesPaint()方法设置图表中的分类图形颜色。

实例 290 生成罗盘图

光盘位置：光盘\MR\10\290

高级

实用指数：★★★

实例说明

Cewolf 和 JFreeChart 组件结合使用时，还可以生成一种有趣的罗盘图，也就是指南针。罗盘中间包含一个指示方向的指针，其边缘则显示出表示不同方向的刻度。本实例将演示如何绘制罗盘图，运行效果如图 10.18 所示。

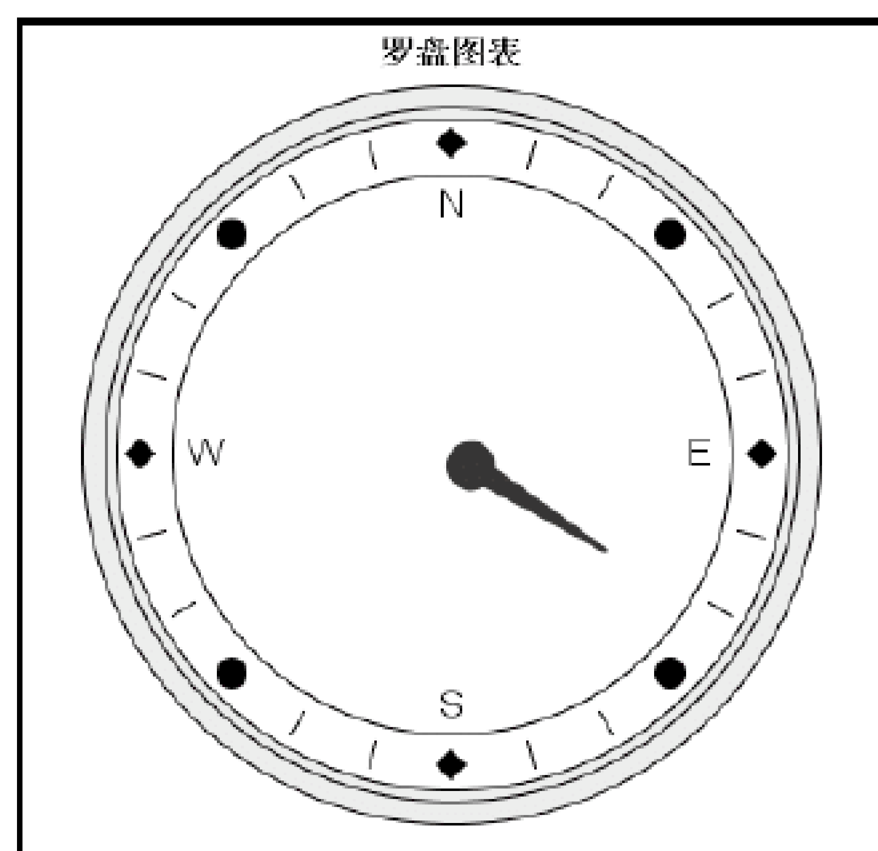


图 10.18　生成罗盘图

关键技术

罗盘图的数据集为 JFreeChart 中的 ValueDateset 接口类型，DefaultValueDateset 类则是 ValueDateset 接口的实现类。创建 DefaultValueDateset 类的对象时，其构造方法需要接收一个双精度类型的参数，该参数表示方位的角度，如 60.0 表示 60°。

JFreeChart 中的 CompassPlot 类是专门用来处理罗盘图的，该类中提供了多个方法用于对图表进行处理，如图表中的字体、指针形状、图表颜色等。

另外，在应用<cewolf:chart>生成罗盘图时，需要将 type 属性设置为 compass。

设计过程

（1）创建 CompassDatasetProducer 类并实现 DatasetProducer 接口，创建罗盘图的数据集。代码如下：

```
public class CompassDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
  public Object produceDataset(Map params)throws DatasetProduceException
   {
        ValueDataset dataset = new DefaultValueDataset(Math.random()*360);
         return dataset;
   }
   public boolean hasExpired(Map params, Date since)
   {
      return false;
   }
   public String getProducerId()
   {
      return "compassdata";
   }
}
```

（2）创建 CompassPostProcessor 类并实现 ChartPostProcessor 接口，对罗盘图进行加工处理。代码如下：

```
public class CompassPostProcessor implements ChartPostProcessor, Serializable
{
    public void processChart (Object chart, Map params) {
```

```
        JFreeChart compassChart = (JFreeChart) chart;
        compassChart.getTitle().setFont(new Font("宋体",Font.BOLD,15));          //标题字体
        compassChart.getLegend().setItemFont(new Font("宋体",Font.BOLD,12));  //分类图例字体
        CompassPlot plot = (CompassPlot)compassChart.getPlot();
        plot.setSeriesNeedle(9);
        plot.setSeriesPaint(0, Color.RED);
        plot.setSeriesOutlinePaint(0, Color.RED);
        plot.setLabelFont(new Font("宋体",Font.BOLD,12));
    }
}
```

秘笈心法

心法领悟 290：设置罗盘的指针形状。

CompassPlot 类的 setSeriesNeedle(int type)方法就是用来设置指针形状的，该方法需要接收一个 int 参数，取值范围为 0～9，这些取值分别代表了不同形状的指针。

实例 291	生成速度图 光盘位置：光盘\MR\10\291	高级 实用指数：★★★

实例说明

利用 Cewolf 还可以生成一个比较有趣的图表，就是用来表示速度的图表。运行本实例，显示出一个速度图，并且在其中根据颜色的不同显示了速度的不同区间，如图 10.19 所示。

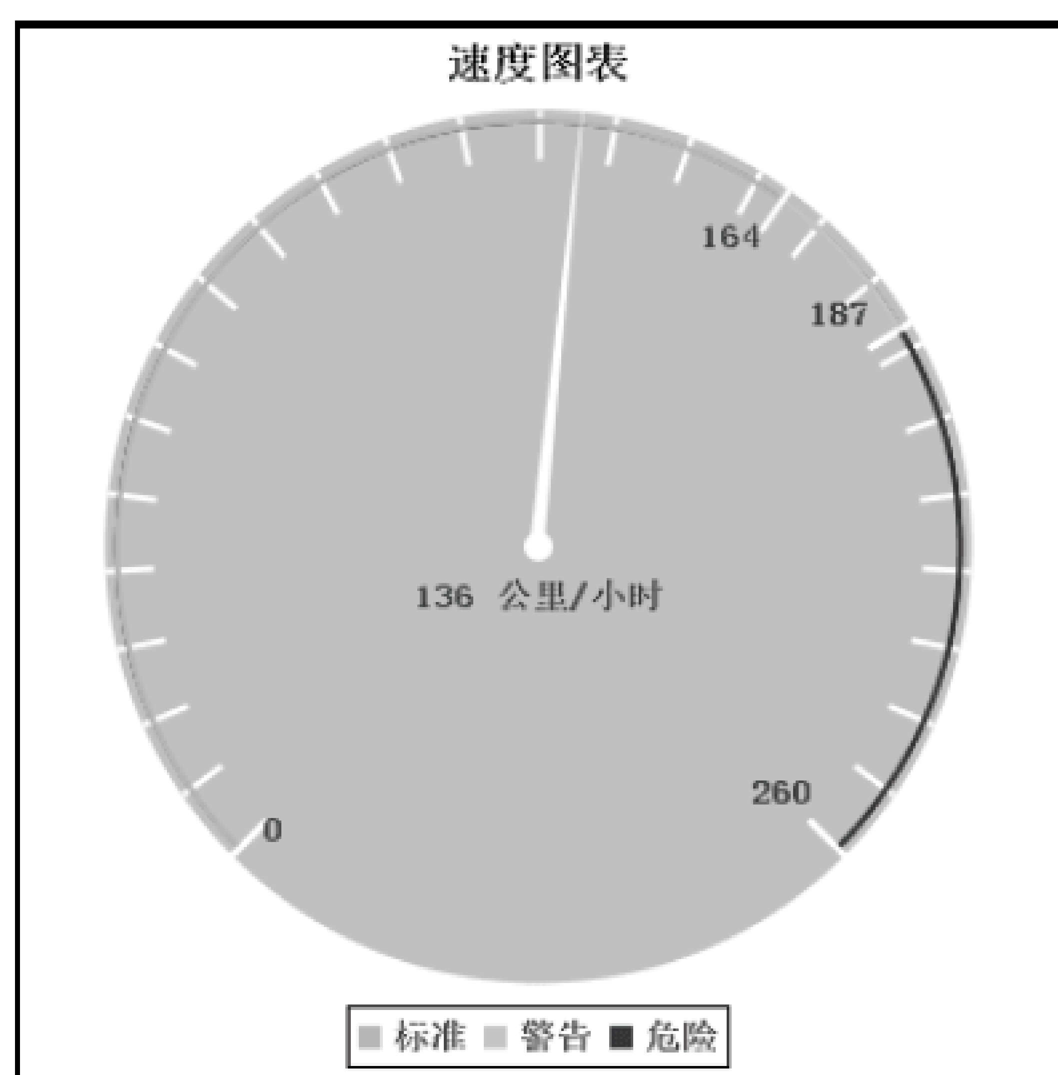


图 10.19　速度图

关键技术

速度图的数据集与罗盘图相同，都是 ValueDateset 接口类型，所以这里还是用 ValueDateset 接口的实现类 DefaultValueDateset 作为速度图的数据集。

JFreeChart 中的 MeterPlot 类是专门用来处理速度图的，该类中提供了多个方法用于对图表进行处理，如图表中的字体、图表不同的速度区间等。

另外，在应用<cewolf:chart>生成速度图时，需要将 type 属性设置为 meter。

（1）创建 MeterDatasetProducer 类并实现 DatasetProducer 接口，创建速度图的数据集。代码如下：

```
public class MeterDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
        double min = 0;
        static double max = 260;
```

```
        public Object produceDataset(Map params)throws DatasetProduceException{
            int speed = 1+(int)(Math.random() * max);
            DefaultValueDataset data = new DefaultValueDataset(speed);
            return data;
        }
    public boolean hasExpired(Map params, Date since){
        return false;
    }
    public String getProducerId(){
        return "meterdata";
    }
}
```

（2）创建 MeterPostProcessor 类并实现 ChartPostProcessor 接口，用于对速度图进行处理。代码如下：

```
public class MeterPostProcessor implements ChartPostProcessor, Serializable{
    public void processChart (Object chart, Map params) {
            double min = 0;
            double max = 260;
            double val = 86;
            double minCrit = 187;
            double maxCrit = max;
            double minWarn = 164;
            double maxWarn = minCrit;
            double maxNorm = minCrit;
            double minNorm = min;
            JFreeChart meterChart = (JFreeChart) chart;
            meterChart.getTitle().setFont(new Font("宋体",Font.BOLD,15));          //标题字体
            meterChart.getLegend().setItemFont(new Font("宋体",Font.BOLD,12));     //分类图例字体
            MeterPlot plot = (MeterPlot) meterChart.getPlot();
            plot.setRange(new Range(min, max));
            plot.addInterval(new MeterInterval("标准", new Range(minNorm, maxNorm),
                                    Color.green, new BasicStroke(2.0f), null));
            plot.addInterval(new MeterInterval("警告", new Range(minWarn, maxWarn),
                                    Color.yellow, new BasicStroke(2.0f), null));
            plot.addInterval(new MeterInterval("危险", new Range(minCrit, maxCrit),
                                    Color.red, new BasicStroke(2.0f), null));
            plot.setValueFont(new Font("宋体",Font.BOLD,12));
            plot.setTickLabelFont(new Font("宋体",Font.BOLD,12));
            plot.setUnits("公里/小时");
        }
}
```

（3）创建 index.jsp 页面，显示生成的速度图。代码如下：

```
<%
 pageContext.setAttribute("dataset", new MeterDatasetProducer());
 pageContext.setAttribute("meterPP", new MeterPostProcessor());
%>
<cewolf:chart
        id="meterChart"
        title="速度图表"
        type="meter">
    <cewolf:data>
        <cewolf:producer id="dataset" />
    </cewolf:data>
   <cewolf:chartpostprocessor id="meterPP">
   </cewolf:chartpostprocessor>
</cewolf:chart>
<cewolf:img  chartid="meterChart" renderer="/cewolf" width="500"   height="360">
</cewolf:img>
```

心法领悟 291：设置速度的区间。

本实例在生成速度图时，根据速度的大小设置了速度的区间，然后通过速度区间来体现出不同的状态。例如，某一速度值可能在标准、警告或危险的区间内。MeterPlot 类的 addInterval(MeterInterval interval)方法用于设置速度的区间范围，即创建 MeterInterval 类的对象来表示一个速度的区间范围。MeterInterval 类的构造方法如下：

```
public MeterInterval(String label ,Range range,Paint outLinePaint,Stroke outLineStroke,Paint backgroundPaint)
```

参数说明

❶ label：设置不同的速度区间名称。

❷ range：设置速度的区间范围。

❸ outLinePaint：设置速度图的速度区间范围的颜色。

❹ outLineStroke：设置速度图的速度区间范围的笔触。

❺ backgroundPaint：设置背景色。

10.7　综合图表的应用

实例 292　利用柱形图对比不同城市的房价

光盘位置：光盘\MR\10\292

高级　实用指数：★★★

实例说明

柱形图也就是前面实例中所讲的直方图。本实例将通过 Cewolf 组件生成的柱形图来分析不同城市（包括北京、上海、长春）的房价走势，运行结果如图 10.20 所示。

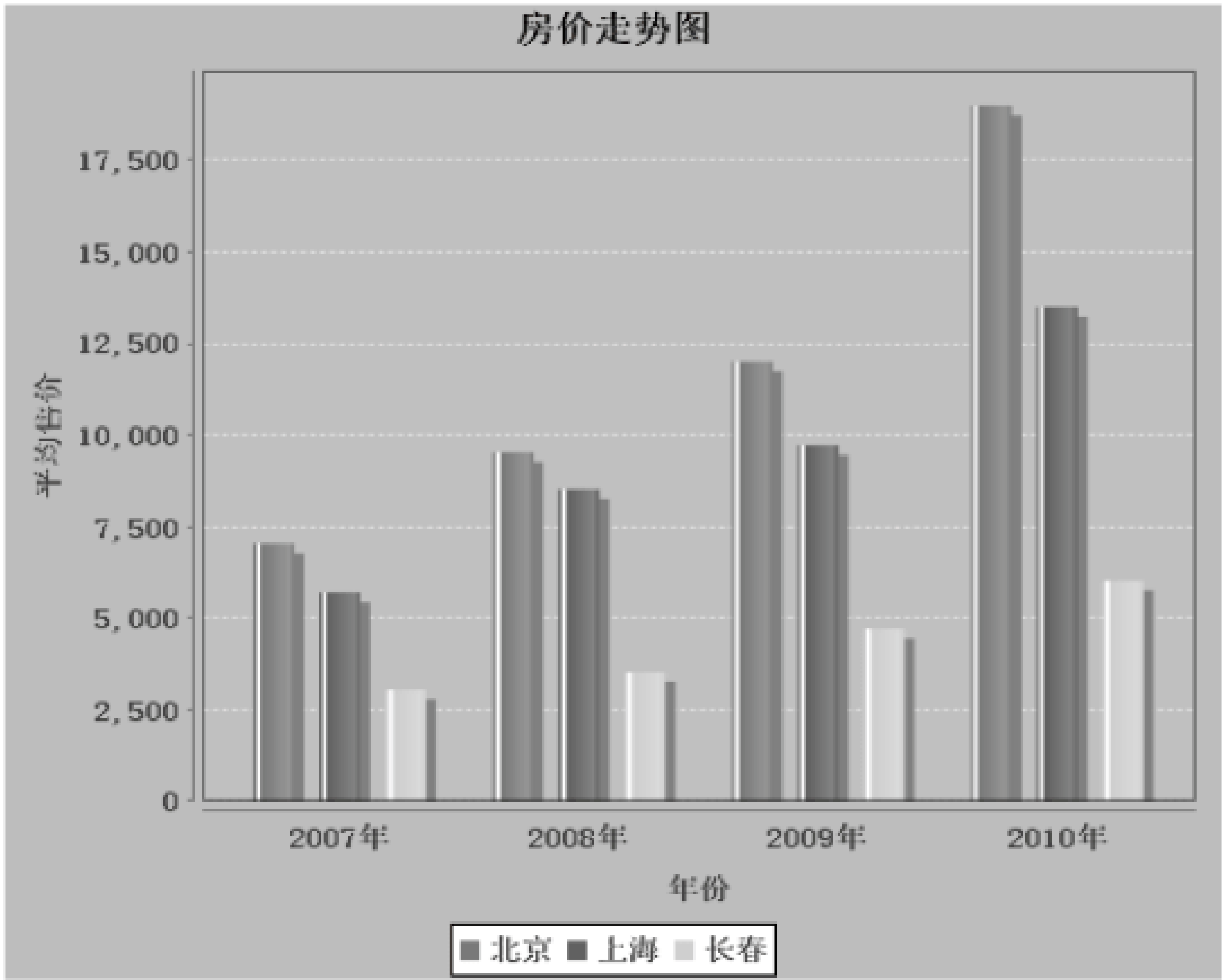
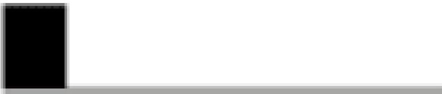


图 10.20　房价走势的柱形图

关键技术

本实例应用的是普通的垂直柱形图，需要设置 CategoryDataset 作为数据集，然后在数据集中添加 3 个城市房价的随机数据即可。

在 JSP 中通过<cewolf:chart>生成垂直柱形图时，需要将 type 属性设置为 verticalBar 类型。



（1）创建 BarDatasetProducer 类并实现 DatasetProducer 接口，创建房价走势柱形图的数据集，并添加房价数据。代码如下：

```
public class BarDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
private    String category[] = {"2007 年", "2008 年", "2009 年", "2010 年" };
public Object produceDataset(Map params){
      DefaultCategoryDataset dataset = new DefaultCategoryDataset();
      int[] priceBJ = {7000,9500,12000,19000};
      int[] priceSH = {5700,8500,9700,13500};
      int[] priceCC = {3000,3500,4700,6000};
```

```
        for (int i = 0; i < category.length; i++){                                    //循环添加数据
                priceBJ[i] = priceBJ[i]+(int)(Math.random() * 50);
                priceSH[i] = priceSH[i]+(int)(Math.random() * 50);
                priceCC[i] = priceCC[i]+(int)(Math.random() * 50);
                dataset.addValue(priceBJ[i],"北京", category[i]);
                dataset.addValue( priceSH[i], "上海", category[i]);
                dataset.addValue(priceCC[i], "长春", category[i]);
        }
        return dataset;
}
public String getProducerId(){
        return "CategoryDataProducer";
}
public boolean hasExpired(Map params, Date since){
        return false;
}
}
```

（2）创建 BarPostProcessor 类并实现 ChartPostProcessor 接口，处理柱形图的字体。代码如下：

```
public class BarPostProcessor implements ChartPostProcessor, Serializable{
        public void processChart (Object chart, Map params) {
                JFreeChart barChart = (JFreeChart) chart;
                barChart.getTitle().setFont(new Font("宋体",Font.BOLD,15));            //标题字体
                barChart.getLegend().setItemFont(new Font("宋体",Font.BOLD,12));       //分类图例字体
                CategoryPlot   plot = (CategoryPlot)barChart.getPlot();
                plot.getDomainAxis().setLabelFont(new Font("宋体",Font.BOLD,12));
                plot.getDomainAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));
                plot.getRangeAxis().setLabelFont(new Font("宋体",Font.BOLD,12));
                plot.getRangeAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));
}
}
```

（3）编写 index.jsp 页面，生成柱形图，显示出房价的走势。代码如下：

```
<%
 pageContext.setAttribute("dataset",new BarDatasetProducer());                      //创建数据集
 pageContext.setAttribute("barPP",new BarPostProcessor());                          //图表样式
%>
   <cewolf:chart type="verticalBar"
                        id="verticalBarChart"
                        title="房价走势图"
                        yaxislabel="平均售价"
                        xaxislabel="年份"
                        backgroundcolor="#99CC99">
        <cewolf:data>
                <cewolf:producer id="dataset" />
        </cewolf:data>
        <cewolf:chartpostprocessor id="barPP"></cewolf:chartpostprocessor>
   </cewolf:chart>
   <cewolf:img chartid="verticalBarChart" height="400" width="500" renderer="/cewolf"></cewolf:img>
```

秘笈心法

心法领悟 292：图例的背景色。

JFreeChart 中的 LegendTitle 类就是用于处理图例的，使用 LegendTitle 类的 setBackgroundPaint(Paint paint)方法可以设置图例的背景色。

实例 293 利用饼图显示投票结果

光盘位置：光盘\MR\10\293

高级

实用指数：★★★

在 Web 程序中经常要用到在线投票栏目，其最终目的是获得投票的统计信息，利用统计信息生成图表能够将统计结果更直观地展示给用户。本实例将演示如何利用饼图来显示投票结果，运行结果如图 10.21 所示。

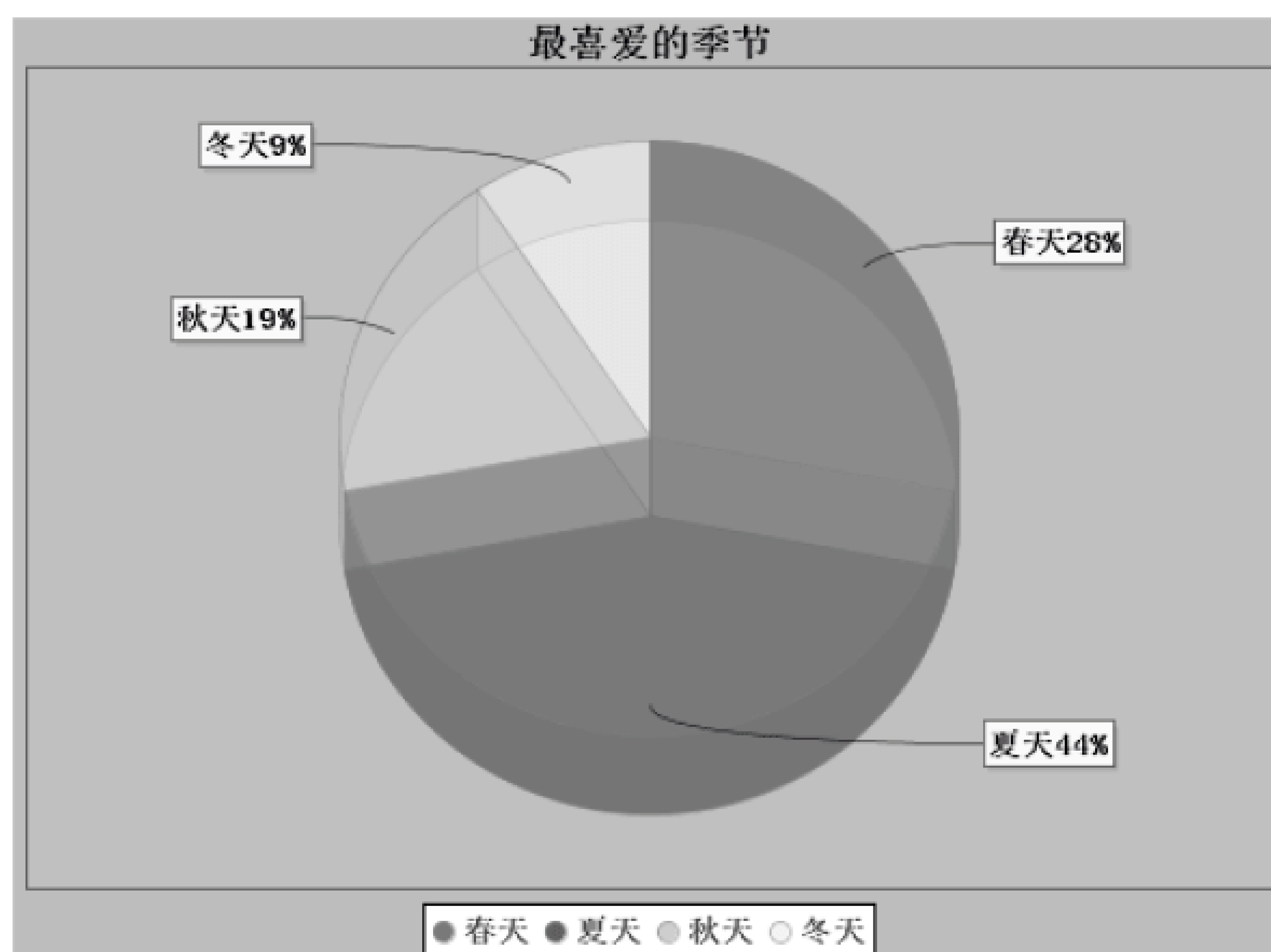


图 10.21　分析投票结果的饼图（3D）

关键技术

本实例的实现比较简单，首先要知道生成饼图所使用的数据集是 PieDataset 类型，DefaultPieDataset 类是 PieDataset 接口的实现，接下来可以创建 DefaultPieDataset 类的对象，然后向该数据集中添加投票数据，最后通过<cewolf:chart>生成投票结果的饼图。

（1）创建 PieDatasetProducer 类并实现 DatasetProducer 接口，用于生成分析投票结果的饼图数据集。代码如下：

```
public class PieDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
 private   String title[] = {"春天", "夏天", "秋天", "冬天"};
 public Object produceDataset(Map params)
   {
      DefaultPieDataset dataset = new DefaultPieDataset();                        //饼图数据集
      int[] count = {240,380,160,80};
      for (int i = 0; i < title.length; i++)
      {
        dataset.setValue(title[i], count[i]);                                      //添加饼图数据
      }
      return dataset;
   }
   public String getProducerId()
   {
      return "PieDataProducer";
   }
   public boolean hasExpired(Map params, Date since)
   {
      return false;
   }
}
```

（2）创建 PiePostProcessor 类并实现 ChartPostProcessor 接口，对饼图的显示样式和字体进行处理。代码如下：

```
public class PiePostProcessor implements ChartPostProcessor, Serializable
{
      public void processChart (Object chart, Map params) {
              JFreeChart pieChart = (JFreeChart) chart;
              pieChart.getTitle().setFont(new Font("宋体",Font.BOLD,15));              //标题字体
              pieChart.getLegend().setItemFont(new Font("宋体",Font.BOLD,12));    //分类图例字体
              PiePlot plot = (PiePlot)pieChart.getPlot();
              plot.setLabelFont(new Font("宋体",Font.BOLD,12));
              plot.setForegroundAlpha(0.5f);
```

```
            plot.setLabelGenerator(
                    new StandardPieSectionLabelGenerator("{0} {2}",
                    NumberFormat.getNumberInstance(),
                    NumberFormat.getPercentInstance()));          //设置分类标签的格式，更改数字的显示格式为百分比
 }
}
```

（3）创建 index.jsp 页，显示生成的图表。代码如下：

```
<%
 pageContext.setAttribute("pieData", new PieDatasetProducer());          //创建数据集
 pageContext.setAttribute("pieTip", new ToolTip());                      //工具提示
 pageContext.setAttribute("piePP", new PiePostProcessor());              //加工饼图
%>
  <cewolf:chart type="pie3D"
                id="pie3DChart"
                title="最喜爱的季节"
                backgroundcolor="#99CC99">
      <cewolf:data >
            <cewolf:producer id="pieData" />
      </cewolf:data>
      <cewolf:chartpostprocessor id="piePP"></cewolf:chartpostprocessor>
  </cewolf:chart>
  <cewolf:img chartid="pie3DChart" height="370" width="500" renderer="/cewolf">
      <cewolf:map tooltipgeneratorid="pieTip"></cewolf:map>
  </cewolf:img>
```

秘笈心法

心法领悟 293：饼图分类标签的百分比数值。

在处理饼图时，可以将饼图中的分类标签数值以百分比的形式显示。StandardPieSectionLabelGenerator 类的构造函数可以重新生成图表标签。在构造函数中传递不同的参数具有不同意义。例如，本实例中在 StandardPieSectionLabelGenerator 的构造函数中传递了参数“{0}{2}”，这个参数表示显示“类别名称”和“类别百分比”的样式。

实例 294 利用折线图分析某城市蔬菜价格走势

光盘位置：光盘\MR\10\294

高级

实用指数：★★★

本实例将介绍如何通过 Cewolf 组件绘制的折线图来分析某城市 3 种蔬菜的价格走势，运行结果如图 10.22 所示。

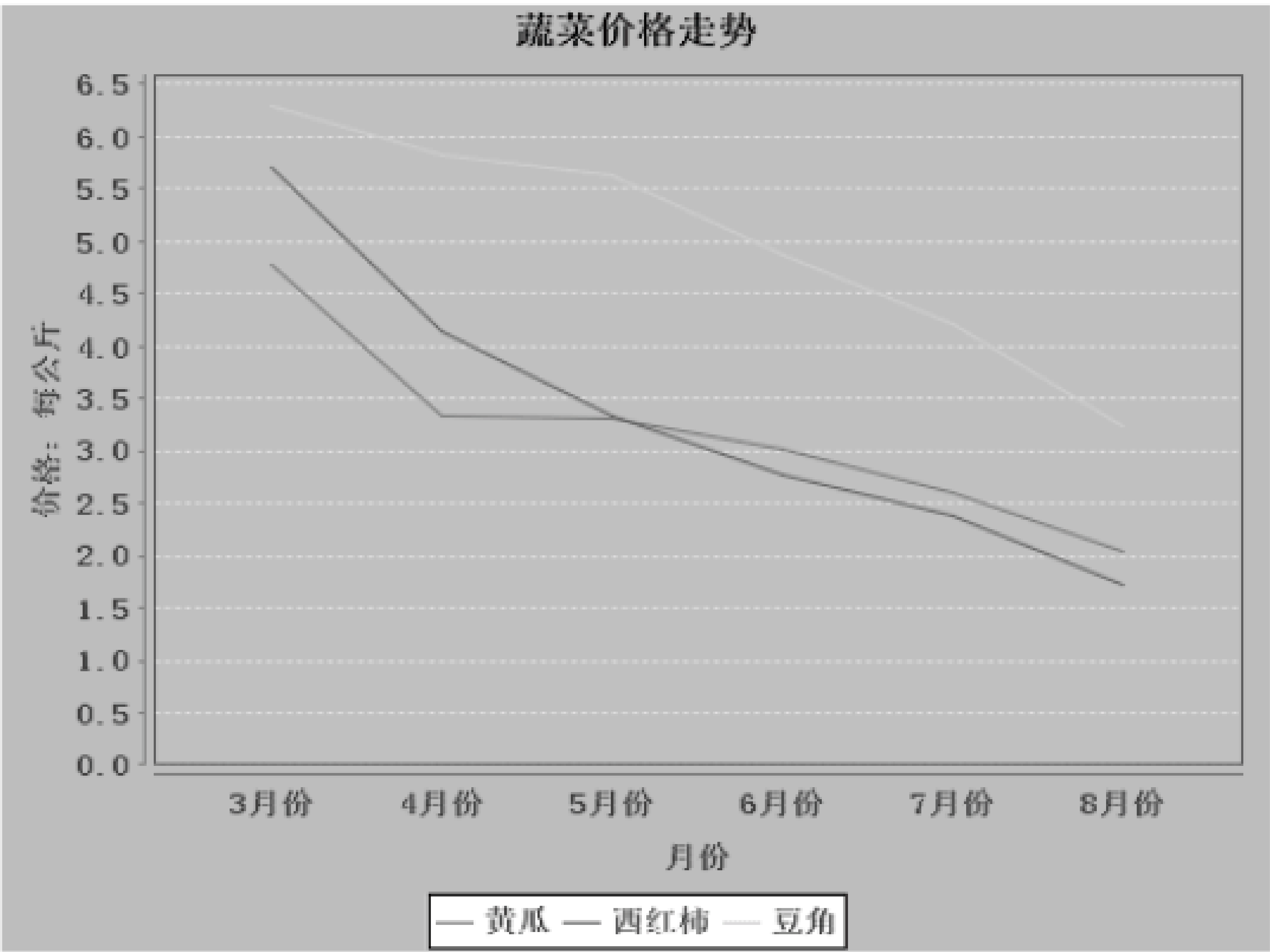


图 10.22　蔬菜的价格走势折线图

关键技术

折线图也就是前面实例中所说的线段图。它有两种类型的图表，即基于 CategoryDataset 数据集的图表和基于 XYDataset 数据集的图表。本实例实现的是基于 CategoryDataset 数据集的图表。

在应用<cewolf:chart>生成基于 CategoryDataset 数据集的图表时，应该将 type 属性设置为 line。

设计过程

（1）创建 LineDatasetProducer 类并实现 DatasetProducer 接口，生成折线图的数据集。代码如下：

```
public class LineDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
  public Object produceDataset(Map params){
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        double[]priceHG = {4.5D,3.3D,3.0D,2.8D,2.5D,1.9D};              //黄瓜价格
        double[]priceXHS={5.5D,3.9D,3.3D,2.6D,2.3D,1.5D};                //西红柿价格
        double[]priceDJ={6.0D,5.8D,5.5D,4.8D,4.0D,3.0D};                 //豆角价格
        for(int i=0;i<6;i++){
              priceHG[i] = priceHG[i]+Math.random()*0.3;
              priceXHS[i] = priceXHS[i]+Math.random()*0.3;
              priceDJ[i] = priceDJ[i]+Math.random()*0.3;
              dataset.addValue(priceHG[i], "黄瓜", (i+3)+"月份");
              dataset.addValue(priceXHS[i], "西红柿", (i+3)+"月份");
              dataset.addValue(priceDJ[i], "豆角", (i+3)+"月份");
        }
     return dataset;
  }
  public String getProducerId(){
     return "lineDataProducer";
  }
  public boolean hasExpired(Map params, Date since){
     return true;
  }
}
```

（2）创建 LinePostProcessor 类并实现 ChartPostProcessor 接口，处理折线图的字体。代码如下：

```
public class LinePostProcessor implements ChartPostProcessor, Serializable
{
     public void processChart (Object chart, Map params) {
            JFreeChart lineChart = (JFreeChart) chart;
            lineChart.getTitle().setFont(new Font("宋体",Font.BOLD,15));          //标题字体
            lineChart.getLegend().setItemFont(new Font("宋体",Font.BOLD,12));     //分类图例字体
            CategoryPlot plot = lineChart.getCategoryPlot();
       plot.getDomainAxis().setLabelFont(new Font("宋体",Font.BOLD,12));
       plot.getDomainAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));
       plot.getRangeAxis().setLabelFont(new Font("宋体",Font.BOLD,12));
       plot.getRangeAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));
 }
}
```

秘笈心法

心法领悟 294：加粗折线。

折线图中默认的折线都比较细，可以根据实际需要修改其笔触。使用 LineAndShapeRenderer 类的 setSeriesStroke()方法即可设置折线图的笔触。

实例 295	利用区域图对比分析员工业绩 光盘位置：光盘\MR\10\295	高级 实用指数：★★★

实例说明

区域图的表现形式是多种多样的，可以用于描述数据的变化程度、部分与整体的关系、数据对比等。本实

例将演示如何利用区域图对比分析员工销售业绩，运行结果如图 10.23 所示。

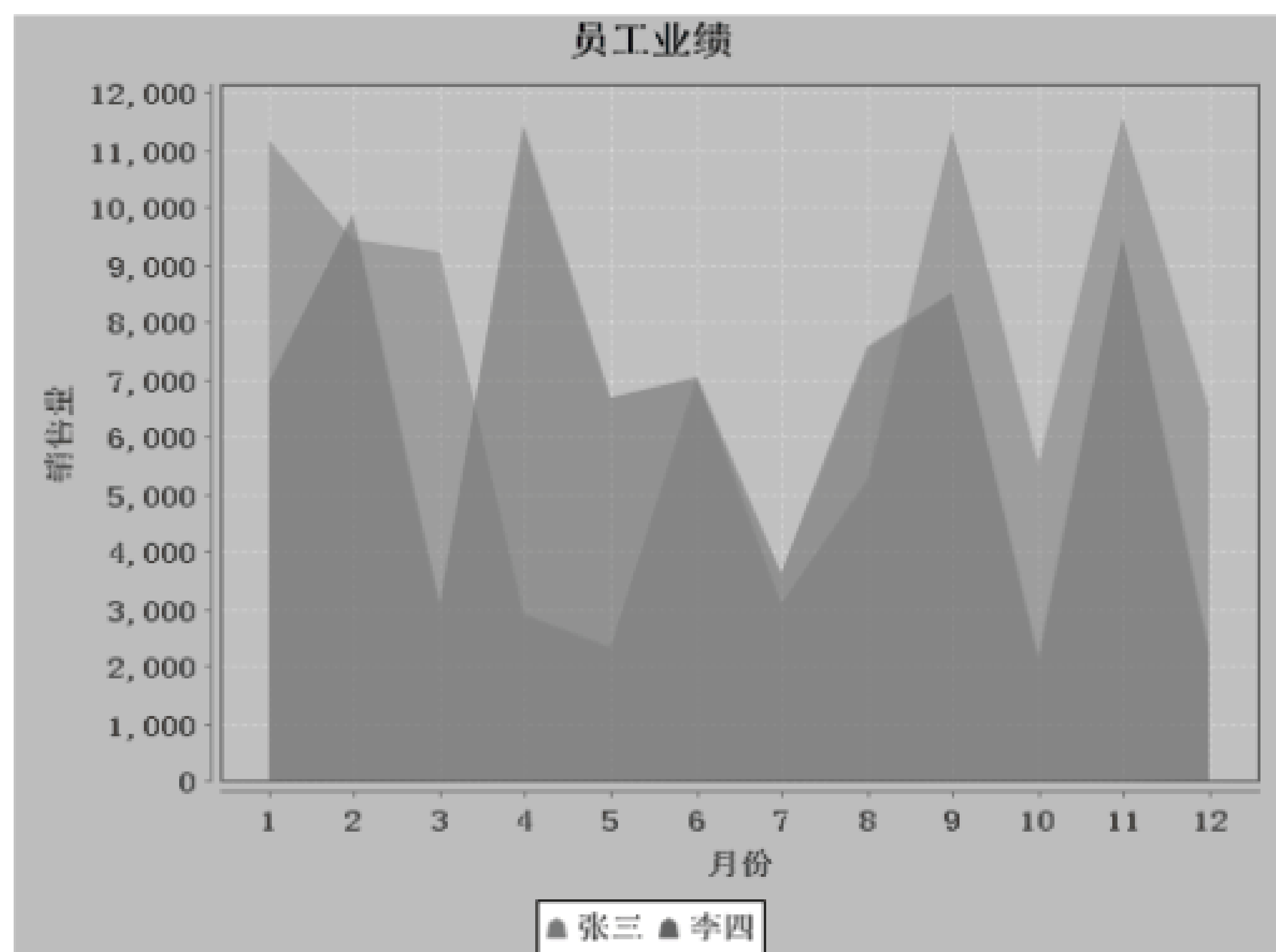


图 10.23　分析员工业绩的区域图

关键技术

在生成区域图时，可以使用 CategoryDataset 或 XYDataset 作为数据集。本实例应用的是 XYDataset。

（1）创建 XYAreaDatasetPRoducer 类并实现 DatasetProducer 接口，用于生成员工业绩的区域图数据集。代码如下：

```
public class XYAreaDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
  public Object produceDataset(Map params){
       XYSeries xysZS = new XYSeries("张三");
       XYSeries xysLS = new XYSeries("李四");
       int salesZS = 0;
       int salesLS = 0;
       for (int month = 1; month <= 12; month++){
          salesZS =new Random().nextInt(10000)+2000;
          salesLS = new Random().nextInt(10000)+2000;
          xysZS.add(month, salesZS);
          xysLS.add(month, salesLS);
       }
       XYSeriesCollection xysc = new XYSeriesCollection();
       xysc.addSeries(xysZS);
       xysc.addSeries(xysLS);
       return xysc;
   }
   public String getProducerId(){
       return "areaDataProducer";
   }
   public boolean hasExpired(Map params, Date since){
       return true;
   }
}
```

（2）创建 XYAreaPostProcessor 类并实现 ChartPostProducer 接口，用于处理图表的字体。代码如下：

```
public class XYAreaPostProcessor implements ChartPostProcessor, Serializable{
      public void processChart (Object chart, Map params) {
              JFreeChart lineChart = (JFreeChart) chart;
              lineChart.getTitle().setFont(new Font("宋体",Font.BOLD,15));             //标题字体
              lineChart.getLegend().setItemFont(new Font("宋体",Font.BOLD,12));        //分类图例字体
              XYPlot plot = lineChart.getXYPlot();
              plot.getDomainAxis().setLabelFont(new Font("宋体",Font.BOLD,12));        //X 轴标题字体
              plot.getDomainAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));    //X 轴刻度线字体
              plot.getRangeAxis().setLabelFont(new Font("宋体",Font.BOLD,12));         //Y 轴标题字体
              plot.getRangeAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));     //Y 轴刻度线字体
}
}
```

（3）创建 index.jsp 页，显示生成的区域图，分析员工的业绩。代码如下：

```
<%
 pageContext.setAttribute("xyAreaDataset", new XYAreaDatasetProducer());//创建数据集
 pageContext.setAttribute("xyAreaPP", new XYAreaPostProcessor());
%>
  <cewolf:chart type="areaxy"
                    id="xyAreaChart"
                    title="员工业绩"
                    xaxislabel="月份"
                    yaxislabel="销售量"
                    backgroundcolor="#99CC99">
      <cewolf:data >
            <cewolf:producer id="xyAreaDataset" />
      </cewolf:data>
      <cewolf:chartpostprocessor id="xyAreaPP"></cewolf:chartpostprocessor>
  </cewolf:chart>
  <cewolf:img chartid="xyAreaChart" height="370" width="500" renderer="/cewolf">
  </cewolf:img>
```

秘笈心法

心法领悟 295：设置 X 轴标签的角度。

利用 ValueAxis 类的 setLabelAngle()方法可以设置区域图 X 轴标签的角度。语法如下：

```
public void setLabelAngle(double angle)
```

参数说明

angle：表示区域图 X 轴的标签旋转角度，其值根据弧度计算。

实例 296	利用时序图分析商品月销售收益 光盘位置：光盘\MR\10\296	高级 实用指数：★★★

实例说明

本实例将介绍如何通过 Cewolf 组件绘制的时序图分析商品月销售收益，运行结果如图 10.24 所示。程序中图表的数据是随机生成的，所以每次刷新页面时，这个时序图都会改变。

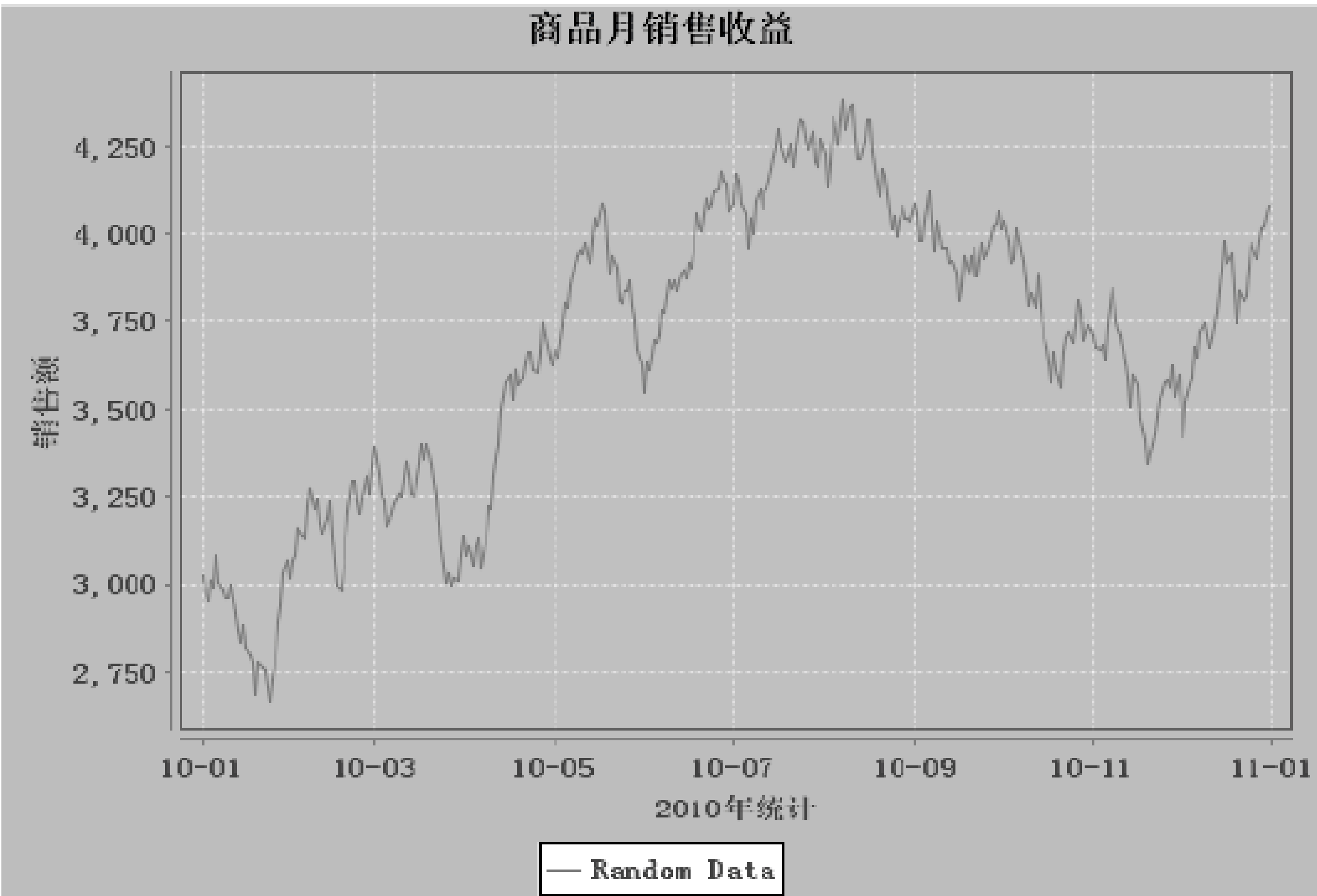


图 10.24　商品月销售收益时序图



时序图的数据集实现类是 TimeSeriesCollection，向 TimeSeriesCollection 数据集中添加数据类型是 TimeSeries 的类对象，然后通过 TimeSeries 的 add()方法添加时序数据。本实例添加的是一年 365 天的数据，所以需要创建 org.jfree.data.time.Day 类的对象来添加表示一年中的每一天。添加一年的数据很简单，只要通过一个 for 循环即可实现，然后在循环中通过 Day 类对象的 next()方法返回指定日期的下一个日期，再添加数据即可。

设计过程

（1）创建 TimeSeriesDatasetProducer 类并实现 DatasetProducer 接口，用于生成一年 365 天的时序数据的数据集。代码如下：

```
public class TimeSeriesDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
        public Object produceDataset(Map params){
                TimeSeries timeseries = new TimeSeries("Random Data");
                Day day = new Day(1, 1, 2010);
                double d = 3000D;
                //添加一年 365 天的数据
                for (int i = 0; i < 365; i++) {
                        d = d + (Math.random() - 0.5) * 200;
                        timeseries.add(day, d);
                        day = (Day) day.next();
                }
                return new TimeSeriesCollection(timeseries);                              //返回数据集合对象
        }
        public String getProducerId()
        {
           return "TimeDataProducer";
        }
        public boolean hasExpired(Map params, Date since)
        {
           return true;
        }
}
```

（2）创建 TimeSeriesPostProcessor 类并实现 ChartPostProcessor 接口，用于处理时序图的字体。代码如下：

```
public class TimeSeriesPostProcessor implements ChartPostProcessor, Serializable{
      public void processChart (Object chart, Map params) {
            JFreeChart lineChart = (JFreeChart) chart;
            lineChart.getTitle().setFont(new Font("宋体",Font.BOLD,15));                  //标题字体
            lineChart.getLegend().setItemFont(new Font("宋体",Font.BOLD,12));             //分类图例字体
            XYPlot plot = lineChart.getXYPlot();
            plot.getDomainAxis().setLabelFont(new Font("宋体",Font.BOLD,12));             //X 轴标题字体
            plot.getDomainAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));         //X 轴刻度线字体
            plot.getRangeAxis().setLabelFont(new Font("宋体",Font.BOLD,12));              //Y 轴标题字体
            plot.getRangeAxis().setTickLabelFont(new Font("宋体",Font.BOLD,12));          //Y 轴刻度线字体
            DateAxis domainAxis = (DateAxis)plot.getDomainAxis();
            SimpleDateFormat format = new SimpleDateFormat("yy-MM");
            domainAxis.setDateFormatOverride(format);
}
}
```

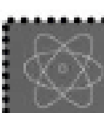
（3）创建 index.jsp 页，显示生成的时序图。代码如下：

```
<%
pageContext.setAttribute("timeDataset", new TimeseriesDatasetProducer());                //创建数据集
pageContext.setAttribute("timeseriesPP", new TimeseriesPostProcessor());
%>
  <cewolf:chart type="timeseries"
                id="timeseriesChart"
                title="商品月销售收益"
                xaxislabel="2010 年统计"
                yaxislabel="销售额"
                backgroundcolor="#99CC99">
      <cewolf:data >
            <cewolf:producer id="timeDataset" />
      </cewolf:data>
      <cewolf:chartpostprocessor id="timeseriesPP"></cewolf:chartpostprocessor>
  </cewolf:chart>
  <cewolf:img chartid="timeseriesChart" height="370" width="550" renderer="/cewolf">
  </cewolf:img>
```

心法领悟 296：动态显示十字标记。

使用 XYPlot 类的 setDomainCrosshairVisible()方法可以显示出时间轴上的标记。语法如下：

```
setDomainCrosshairVisible(boolean flag)
```



实例 297　利用组合图表分析国际原油价格走势　高级

光盘位置：光盘\MR\10\297　实用指数：★★★

■ 实例说明

本实例将通过 Cewolf 组件生成的组合图表来分析 2009 年和 2010 年每天的国际原油价格走势，运行结果如图 10.25 所示。其中，时序图分析的是 2009 年的价格，直方图分析的是 2010 年的价格。

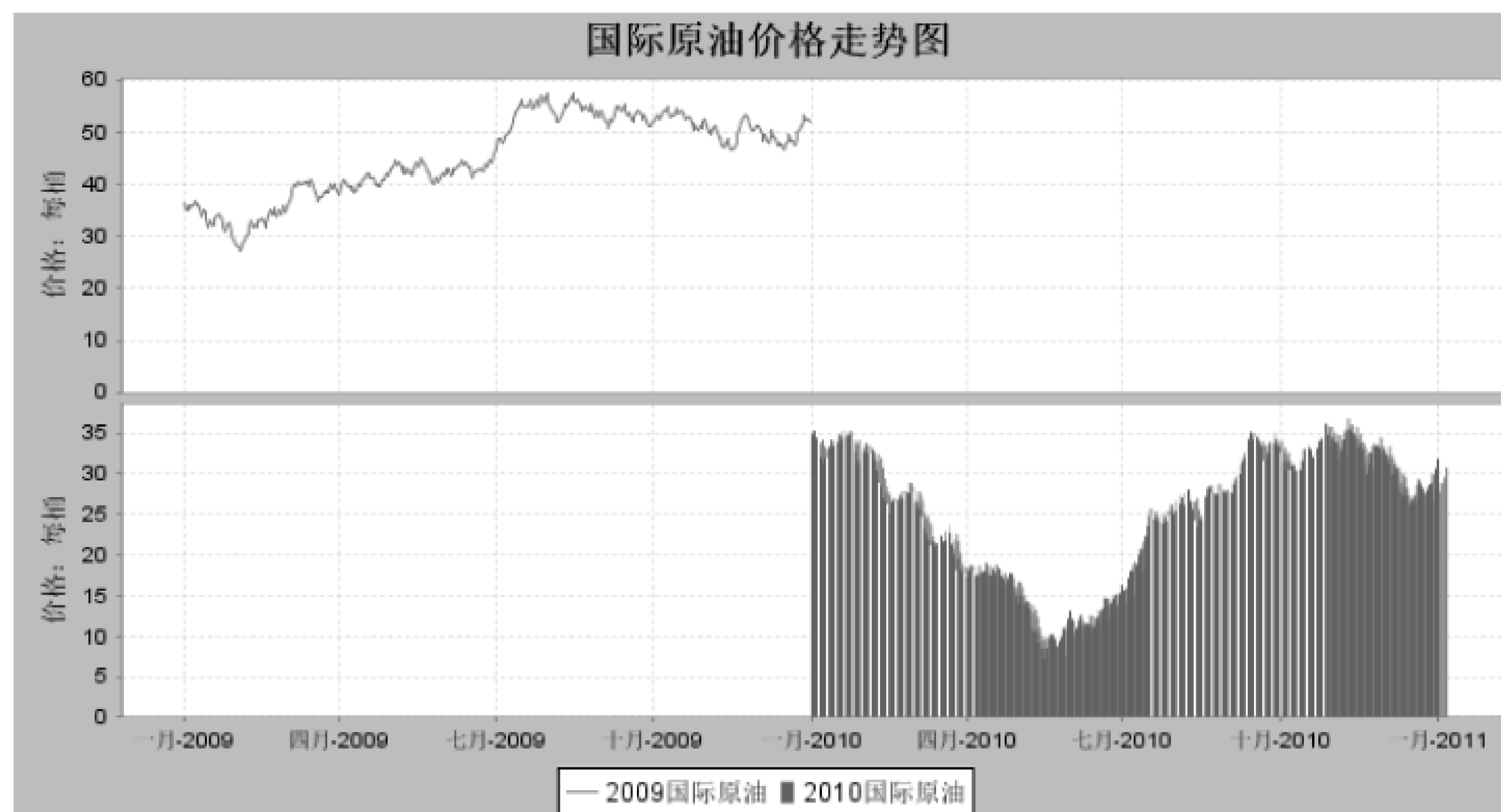


图 10.25　国际原油价格的组合图表

■ 关键技术

在本实例的组合图表中包含了一个时序图和一个直方图，所使用的数据集都是 TimeSeriesCollection，通过<cewolf:combinedchart>来展示组合图表。这里需要通过<cewolf:combinedchart>的子标签<cewolf:plot>来指定时序图的 plot 和直方图的 plot。

■

（1）创建 XYLineDatasetProducer 并实现 DatasetProducer 接口，创建组合图表中时序图的数据集。代码如下：

```
public class XYLineDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
  public Object produceDataset(Map params)
   {
        TimeSeries timeseries = new TimeSeries("2009 国际原油");
              Day day = new Day(1, 1, 2009);
              double value = 35;
              //添加一年 365 天的数据
              for (int i = 0; i < 365; i++) {
                      double flag = Math.random();
                      if(flag>0.5){
                              value=value+Math.random()*1.5;

                      }else{
                              value=value-Math.random()*1.5;
                      }
                      timeseries.add(day, value);
                      day = (Day) day.next();
              }
              //返回数据集合对象
              return new TimeSeriesCollection(timeseries);
   }
   public String getProducerId()
```

```
    {
        return "TimeDataProducer";
    }
    public boolean hasExpired(Map params, Date since)
    {
        return true;
    }
}
```

（2）创建 XYBarDatasetProducer 类并实现 DatasetProducer 接口，生成组合图表中直方图的数据集。代码如下：

```
public class XYBarDatasetProducer implements de.laures.cewolf.DatasetProducer,Serializable {
  public Object produceDataset(Map params){
          TimeSeries timeseries = new TimeSeries("2010 国际原油");
                Day day = new Day(1, 1, 2010);
                double value = 35;
                //添加一年 365 天的数据
                for (int i = 0; i < 365; i++) {
                        double flag = Math.random();
                        if(flag>0.5){
                                value=value+Math.random()*1.5;
                        }else{
                                value=value-Math.random()*1.5;
                        }
                        timeseries.add(day, value);
                        day = (Day) day.next();
                }
                //返回数据集合对象
                return new TimeSeriesCollection(timeseries);
    }
    public String getProducerId(){
        return "TimeDataProducer";
    }
    public boolean hasExpired(Map params, Date since){
        return true;
    }
}
```

（3）创建 index.jsp 页，显示生成的组合图表。代码如下：

```
<%
 pageContext.setAttribute("barDataset", new XYBarDatasetProducer());
 pageContext.setAttribute("lineDataset", new XYLineDatasetProducer());
%>
<cewolf:combinedchart
        id="combinedChart"
        layout="vertical"
        type="combinedxy"
        title="国际原油价格走势图">
      <cewolf:colorpaint color="#99CC99"/>
        <cewolf:plot type="xyline" xaxislabel="time" yaxislabel="价格：每桶">
                <cewolf:data>
                        <cewolf:producer id="lineDataset" />
                </cewolf:data>
        </cewolf:plot>
        <cewolf:plot type="xyverticalbar" xaxislabel="time" yaxislabel="价格：每桶">
                <cewolf:data>
                        <cewolf:producer id="barDataset" />
                </cewolf:data>
        </cewolf:plot>
 </cewolf:combinedchart>
 <cewolf:img chartid="combinedChart" renderer="/cewolf" width="700" height="375"/>
```

心法领悟 297：组合图表的 Y 轴标签。

通过<cewolf:combinedchart>生成组合图表时，即使设置了 yaxislabel 属性值，也不会显示 Y 轴标签。因为组合图表的 Y 轴上有两个图表的 Y 轴数据，所以这里需要通过<cewolf:plot>的 yaxislabel 属性来为每个图表设置 Y 轴标签。

第4篇

Ajax 框架应用篇

⏭ 第 11 章　Prototype 框架

⏭ 第 12 章　jQuery 框架

⏭ 第 13 章　Dojo 框架

第 11 章

Prototype 框架

- 使用 Prototype 基本函数
- Prototype 自定义对象和类
- 对 Ajax 的支持

11.1 使用 Prototype 基本函数

实例 298	使用$()函数获取页面元素 光盘位置：光盘\MR\11\298	高级 实用指数：★★★★

实例说明

Prototype 是一款优秀的 JavaScript 框架和类库，通过它可以更加方便、快捷地开发 JavaScript 应用。此外，Prototype 还提供了对 Ajax 的强大支持，可以非常高效地开发 Ajax 应用程序。Prototype 框架提供了一些非常实用的函数来简化 JavaScript 脚本，如本实例将使用$()函数来获取页面中的元素值，运行结果如图 11.1 所示。

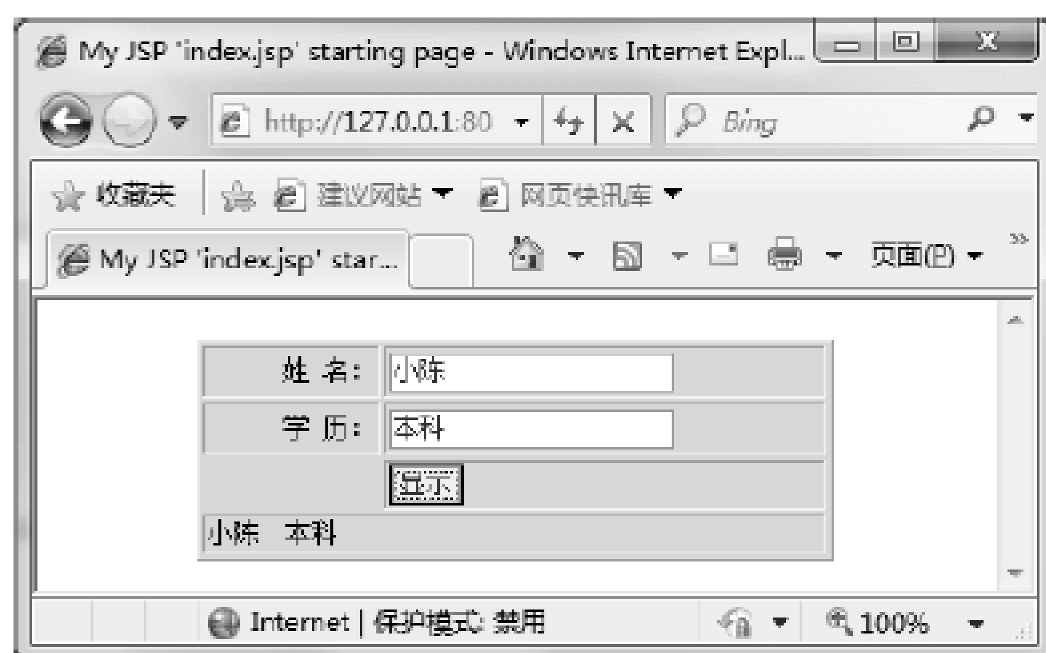


图 11.1 获取页面元素

关键技术

本书以 Prototype 1.7 为基础介绍 Prototype 的使用方法。Prototype 的官方网站为 http://www.prototypejs.org，其中提供了 Prototype 的下载链接（Prototype 框架就是一个名为 prototype.js 的文件）。要想在应用程序中使用 Prototype 框架，不需要设置环境变量，也不需要任何配置文件，就像使用其他的 JS 文件一样，在 JSP 文件的<head></head>标签之间将 prototype.js 文件引入即可。代码如下：

```
<script type="text/javascript" src="JS/prototype.js"></script>
```

**说明：**本实例将 prototype.js 文件放置在 JS 文件夹中。

本实例应用$()函数来获取表单元素值，该函数实现了 Document 对象的 getElementById()方法，具体语法如下。

语法一：获取页面中的一个元素值。语法如下：

```
$(String id);
```

参数说明

id：一个字符串，表示 HTML 元素中的 id 属性值。

函数返回与 id 属性值相匹配的 HTMLElement 对象。

语法二：获取多个HTML元素。语法如下：

```
$(String id1,String id2,…,String idn)
```

参数说明

id1～idn：要获取的 HTML 元素的 id 属性值。

函数返回包含所有 HTML 元素对象的数组。

（1）在 index.jsp 页面中定义两个文本框，id 属性分别为 nameTextfield 与 sageTextfield，然后定义 id 为 out 的<div>层，以及一个标签内容为“显示”的按钮。

（2）在该页面中定义 JavaScript 函数 clickHandler()，用于获取表单中值，并显示在<div>层中。具体代码如下：

```
<script type="text/javascript">
        function clickHandler(){                                        //定义函数
                var name = $('nameTextfield','sageTextfield');          //获取表单元素值
                for(i = 0;i<name.length;i++){                           //循环遍历结果数组
                        $('out').innerHTML += name[i].value+"  ";  //将表单元素显示在<div>层中
                }
                return false;
        }
  </script>
```

秘笈心法

心法领悟 298：注意 id 的唯一性。

$()函数是通过表单元素的 id 属性值来获取表单元素值的，因此保证页面元素的 id 属性的唯一性是很重要的，否则无法预期效果。

实例 299　使用$A()函数实现将参数转换为数组

光盘位置：光盘\MR\11\299

高级

实用指数：★★★★

实例说明

$A()函数用于把单个参数转换成一个 Array 对象。本实例使用$A()函数将页面中的 HTML 节点内容遍历在页面中显示，运行结果如图 11.2 所示。

图 11.2　使用$A()函数实现将参数转换为数组

关键技术

利用$A()函数可以将单个参数转换成一个 Array 对象。结合被 Prototype.js 扩展后的 Array 类，能够方便地把任何可枚举列表转换成或复制到一个 Array 对象，从而更有效地进行遍历。使用$A()函数时，一种推荐的用法就是把 DOM 节点转换成一个普通的 Array 对象。$A()函数的基本语法如下：

```
$A(list)
```

参数说明

list：任意类似数组的集合的引用。

该函数返回一个与之等价的 Array 对象。

注意：如果$A()函数的参数是一个普通变量，而不是一个集合，将返回一个空数组。

（1）在 index.jsp 页面中定义表单，添加文本框、复选框等组件，然后定义<div>层，在该层中添加复选框组件。具体代码如下：

```
<div align="left" id = "loves">
<label>
<input type="checkbox" name="checkbox" value="羽毛球 " />羽毛球
<input type="checkbox" name="checkbox" value="乒乓球 " />乒乓球
<input type="checkbox" name="checkbox" value="上网 " />
</label>上网 <label><br/>
<input type="checkbox" name="checkbox" value="看书 " />
</label>看书
<label>
<input type="checkbox" name="checkbox" value="旅游 " />
</label>旅游 <label>
<input type="checkbox" name="checkbox" value="健身" />健身
<br/>
<input type="checkbox" name="checkbox" value="瑜伽" />瑜伽</label>
</div>
```

（2）在 index.jsp 页面中定义 JavaScript 函数，调用该函数，会将复选框中的值显示在页面中。具体代码如下：

```
<script type="text/javascript">
    function myCheck(){
        var fileList = document.getElementsByName("checkbox");           //获取所有复选框的值
        var fileArray = $A(fileList);                                     //将变量转换为 Array 对象
        for(var i = 0;i<fileArray.length;i++){                           //循环遍历数组对象
                $("view").innerHTML += fileArray[i].value+" ";      //将值在<div>层中显示
```

```
        }
        return false;
    }
</script>
```

■ 秘笈心法

心法领悟 299：$A()函数操作字符串。

借助于 prototype.js 类库扩展后的 Array 类，可以很方便地操作$A()函数返回的数组。不仅如此，$A()函数还可以操作字符串，并将字符串中的每个字符作为数组元素。



实例 300　使用$F()函数获取表单输入控件的值

光盘位置：光盘\MR\11\300　　高级　实用指数：★★★★

■ 实例说明

$F()函数用于获取表单输入控件的值，例如，文本框、文本域、下拉列表框等。该函数与$()函数要区分开，$()函数返回的是 HTML 元素本身，而$F()函数则用于获取表单域的值，而不是表达它本身。本实例实现的是将用户添加的留言标题与留言内容在页面中显示，运行结果如图 11.3 所示。

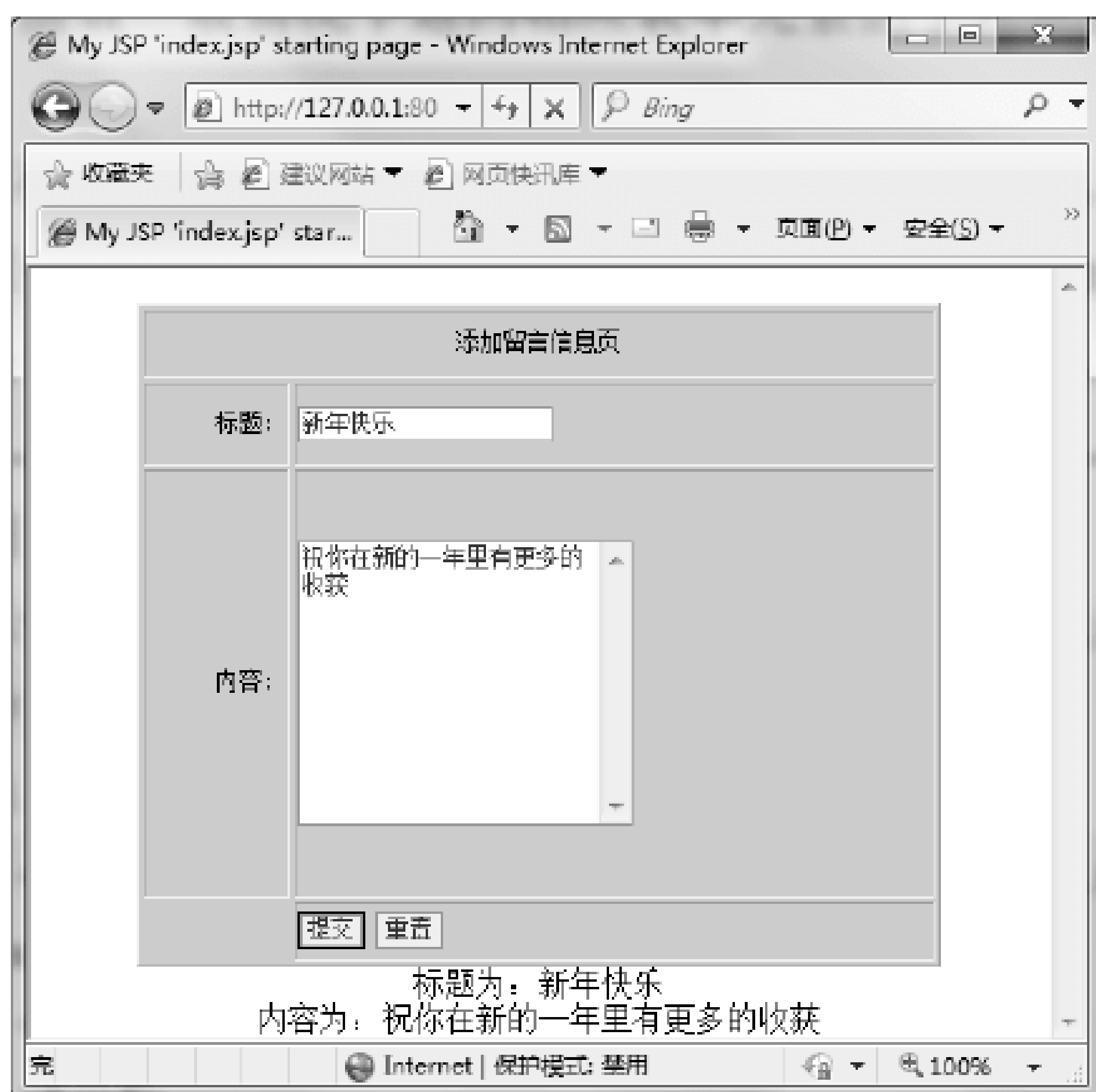


图 11.3　使用$F()函数获取表单输入控件的值

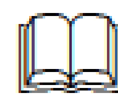
■ 关键技术

$F()函数用于获取任何表单元素的值。语法如下：

```
$F(String id)
```

参数说明

id：一个字符串，表示表单元素的 id 属性值。$F()元素不要求要访问的表单元素必须在表单之内，也可以是表单之外的元素。

说明：$F()函数与$()函数有一个共同点，当在 IE 中使用$F()函数时，该函数不仅可以根据 HTML 元素的 id 属性访问，还可以根据 HTML 元素的 name 属性访问。如果 HTML 元素的 id 属性和 name 属性不一致，则可能导致错误。

（1）在页面中添加表单，并添加 name 属性为 titleTextfield 的文本框、name 属性为 contentTextarea 的文本域。

（2）定义名为 myCheck()的 JavaScript 函数，实现将用户添加的文本框与文本域的值显示在页面的 div 层

中。具体代码如下：

```
<script type="text/javascript">
	function myCheck(){
		var title = $F("titleTextfield");                                    //获取添加的留言标题
		var content = $F("contentTextarea");                                 //获取添加的留言内容
		view.innerHTML = "标题为："+title+"<br>"+"  内容为："+content;        //将留言内容与留言标题在<div>层中显示
		return false;
	}
</script>
```

秘笈心法

心法领悟 300：$F()函数的注意事项。

$F()函数不要求返回的表单元素处于 Form 元素内，因此根本不管页面中有多少个表单元素，它只负责返回页面中第一个满足条件的表单元素的值，也不管该页面元素处于哪个 Form 元素内。

实例 301 使用 Try.these()函数获取返回值

光盘位置：光盘\MR\11\301

中级

实用指数：★★★☆

实例说明

在程序开发中可能会遇到不知调用哪个方法会返回正确结果的情况，这时便可以使用 Try.these()函数。Try.these()函数允许传入一系列函数作为参数。本实例将使用 Try.these()函数创建 XMLHttpRequest 对象并使用变量保存创建的方式，运行结果如图 11.4 所示。

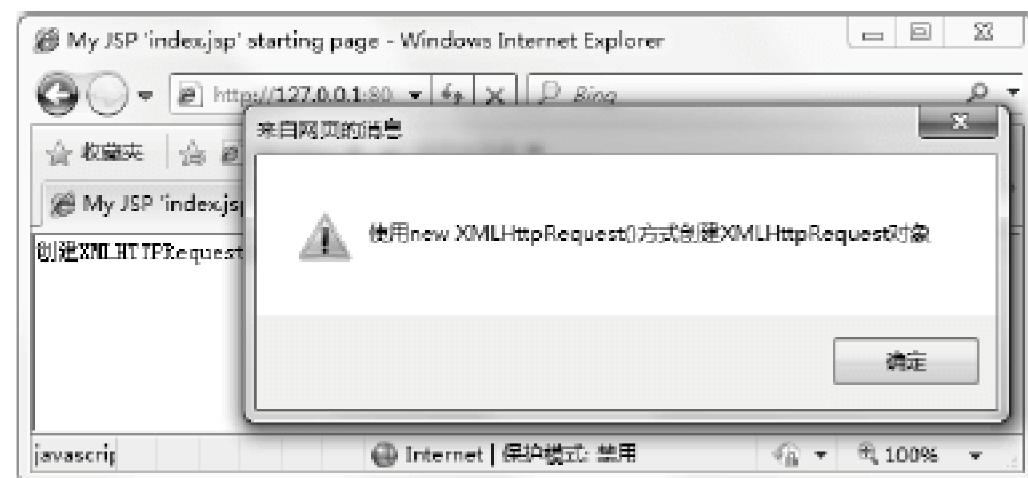


图 11.4　使用 Try.these()函数获取返回值

关键技术

类似于 try…catch 语句，Try.these()函数把一系列的函数作为参数并且按顺序一个一个地执行，直到其中的一个函数成功执行，返回值为成功执行的那个函数的返回值。语法如下：

```
Try.these(function1,function2,…,functionN);
```

参数说明

function1,function2,…,functionN：表示函数的引用。此函数将返回第一个能够成功执行的函数的返回值，如果没有一个函数正确返回则返回 undefined。

（1）定义 JavaScript 函数，在该函数中定义变量 str，用于保存创建 XMLHttpRequest 对象的方式。在函数体中使用 3 种方式创建 XMLHttpRequest 对象，并分别为 str 变量赋予不同的值。具体代码如下：

```
<script type="text/javascript">
	var str = "";                                         //定义变量
	function createXHR(){
		return Try.these(
			function(){
				str = 'new XMLHttpRequest()';         //依次为 str 变量赋值
				return new XMLHttpRequest();          //创建 XMLHttpRequest 对象
			},
			function(){
				str = 'new ActiveXObject("Msxml2.XMLHTTP")';
				return new ActionXObject('Msxml2.XMLHTTP');
			},
			function(){
				str ='new ActiveXObject("Microsoft.XMLHTTP")';
				return new ActionXObject('Microsoft.XMLHTTP');
			}
```

```
            )
        }
</script>
```

（2）在页面中定义超链接，并调用 createXHR()函数。具体代码如下：

```
<body>
    <a href='javascript:if(createXHR())alert("使用"+str+"方式创建 XMLHttpRequest 对象");else alert("创建 XMLHttpRequest 对象失败!")'>创建
XMLHTTPRequest 对象</a>
  </body>
```

秘笈心法

心法领悟 301：Try.these()函数的作用。

Try.these()函数的作用是满足 JavaScript 在不同浏览器上运行的需要。JavaScript 函数在不同的浏览器中可能有不同的结果，有时甚至无法成功运行。为了解决 JavaScript 跨浏览器的问题，经常使用 Try.these()函数来实现效果。

11.2　Prototype 自定义对象和类

实例 302	在 HTML 元素中增加 CSS 样式 光盘位置：光盘\MR\11\302	高级 实用指数：★★★★

实例说明

Prototype.js 提供了大量的自定义对象和类，以简化 JavaScript 开发。本实例使用 Element 对象实现了在 HTML 元素中添加 CSS 样式。运行程序，在如图 11.5 所示页面中单击“增加立体效果”按钮，即可为 HTML 元素中的<div>层添加立体效果，如图 11.6 所示。

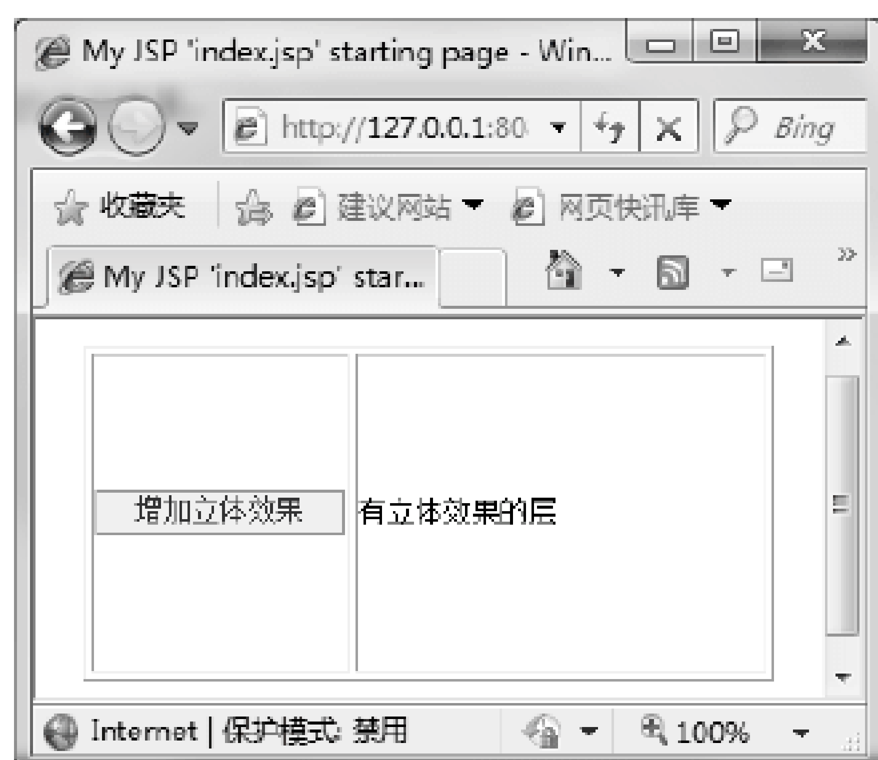


图 11.5　本实例运行首页

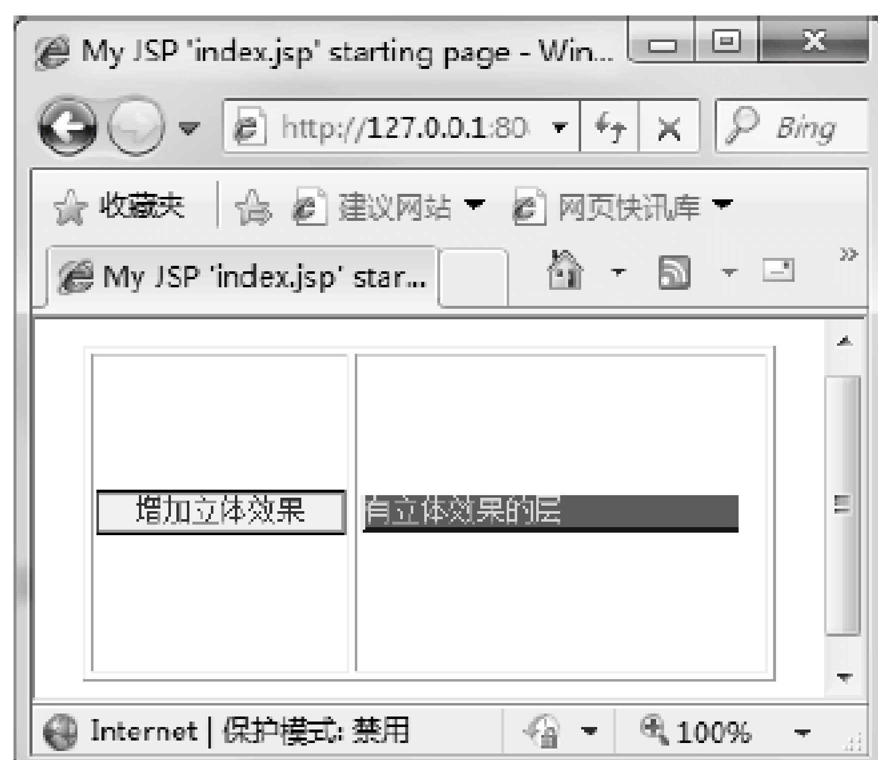


图 11.6　添加了 CSS 样式

Element 对象提供了一系列用于简化 HTML 元素的操作，如通过 CSS 改变 HTML 元素的外观，或直接通过一些方法为 HTML 元素提供动态显示效果。Element 类的常用方法如表 11.1 所示。

表 11.1　Element 类的常用方法

方　法	说　明
addClassName(element,className)	参数 element 表示元素对象引用或元素的 id，参数 calssName 为类选择器名称为元素 element 添加指定的 CSS 样式
classNames(element)	返回一个描述给定对象 CSS 类选择器名称的 Element.ClassNames 对象。参数 element 为元素对象或元素 id
empty(element)	检查一个指示元素标签是否为空（或只含有空格）。参数 element 既可以是元素的 id 属性，也可以是元素本身

续表

方　法	说　明
getHeight(element)	返回元素的 offsetHeight 值。参数 element 为元素对象或元素的 id
getStyle(element,cssProperty)	该方法返回给定元素的 CSS 属性值，无此属性则返回 null。参数 element 为元素对象或元素 id，参数 cssProperty 为 CSS 属性名
inspect(element)	返回值为一个描述元素的格式良好的字符串。参数 element 为元素对象或元素 id
match(element,selector)	检查元素是否匹配给定的CSS选择器。参数element为元素对象或元素id，参数selector为选择器名称
remove(element)	从 Document 对象中移除元素。参数 element 为元素对象或元素 id
visible(element)	检查元素是否可见，如果可见则返回 true，如果不可见则返回 false

设计过程

（1）在项目的 index.jsp 页面中添加表格，在表格中添加按钮和<div>层。具体代码如下：

```
<table width="293" height="142" border="1" align="center">
  <tr>
    <td width="59"><input type="button" onclick="chg()" value="增加立体效果"></td>
    <td width="218"> <div id= "up">有立体效果的层</div></td>
  </tr>
</table>
```

（2）在 index.jsp 页面中定义 JavaScript 函数，当调用该函数时，会为 id 属性为 up 的页面元素添加 CSS 样式。具体代码如下：

```
<script type="text/javascript">
        function chg(){
 Element.addClassName("up","solid");
        }
</script>
```

（3）在页面中定义 CSS 样式，实现立体效果。具体代码如下：

```
<style type="text/css">
        .solid{
                width:160px;
                text-align: conter;
                border-right: "#FFCCCC" 2px solid;
                border-top: "#FFCCCC" 2px solid;
                border-left: #b9ffb9 2px solid;
                color: "#FFCCCC";
                border-bottom: #002200 2px solid;
                background-color: #008000;
        }
 </style>
```

秘笈心法

心法领悟 302：在 CSS 样式中使用字体。

很多设计者喜欢使用各种各样的字体来给页面添彩，但一些字体在大多数用户的机器上都没有安装，因此一定要设置多个备选字体，以避免浏览器直接替换默认的字体。

实例 303　利用 Enumerable 对象在页面中显示数组元素　中级

光盘位置：光盘\MR\11\303　实用指数：★★★★

Enumerable 类是 prototytpe.js 的另一个功能强大的自定义类，该类包含一系列方法，这些方法可以非常方便地遍历 Enumerable 对象中的元素。本实例实现了将在 JavaScript 函数中定义的数组中的元素，以及各个数组中

元素的下标都显示在页面中，运行结果如图 11.7 所示。

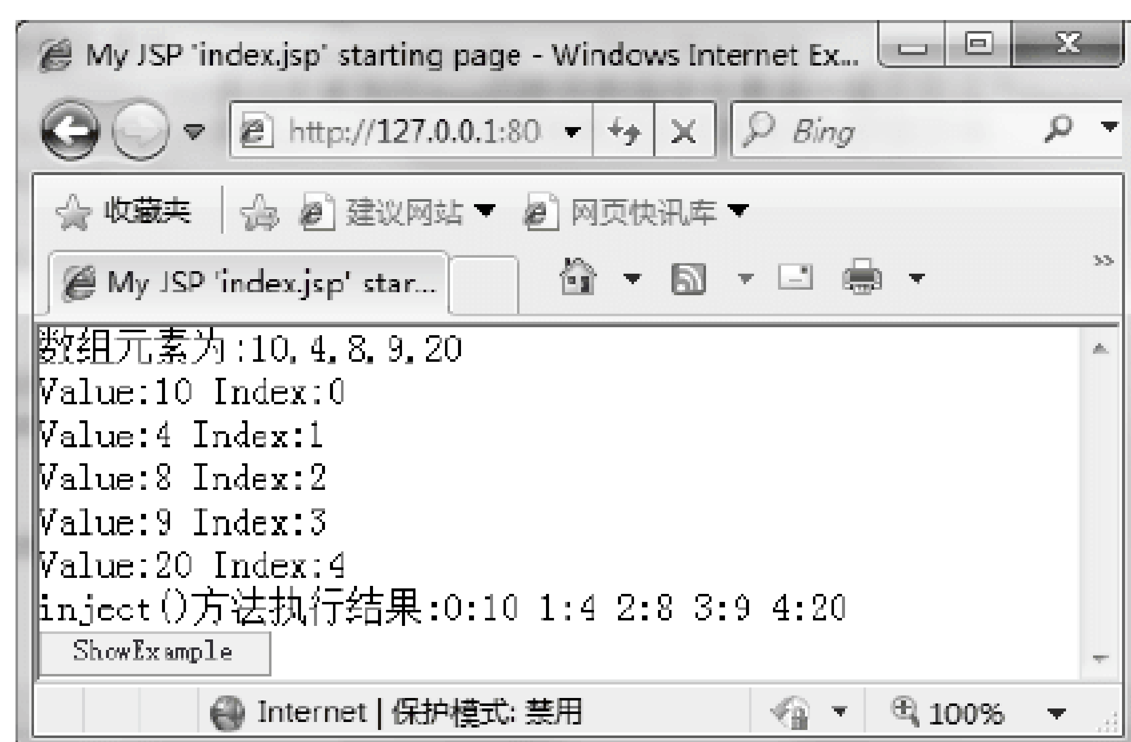


图 11.7　利用 Enumerable 对象在页面中显示数组元素

Enumerable 对象提供了大量方法用于枚举，其常用方法如表 11.2 所示。

表 11.2　Enumerable 类的常用方法

方　　法	说　　明
each(iterator)	反复调用给定的 iterator。其中，参数 iterator 是一个形如 function(value,index)的函数，在 function(value,index)中 value 表示 Enumerable 对象中的元素；index 是集合中元素的索引
collect(iterator)	对集合中的每个元素调用 iterator 并将结果收到数组中返回
any(iterator)	用于测试集合中是否包含任一元素满足某个条件。该函数会用给出的 iterator 测试整个集合
detect(iterator)	用于获取集合中第一个满足某个条件的元素。该函数会使用 iterator 依次处理集合中的每个元素
include(obj)	尝试在集合中查找给定参数的对象。对象被找到则返回 true，否则返回 false
zip(collection1,collection2,…,collectionN)	将给定的集合与当前集合合并。合并操作返回一个元素个数和当前集合相同的新数组

设计过程

在项目的 index.jsp 页面中，定义数组，并将数组中的元素与索引在页面中显示。具体代码如下：

```
<SCRIPT LANGUAGE="JavaScript">
        //Enumerable 对象方法中的 iterator 的前两个参数迭代时将被传递为值与索引
        function foo(v,i){
                $("Result").innerHTML+="Value:"+v+" Index:"+i+"<br>";
                return true;
        }
        //foo1 是为 inject()方法定制的，该方法具有 3 个参数
        function foo1(a,v,i){
                a+=i+":"+v+" ";
                return a;
        }
        function ShowExample(){
                var testArr=[10,4,8,9,20];                              //测试数组
                $("Result").innerHTML="数组元素为:"+testArr+"<br>";      //列出测试数组
                testArr.each(foo);                                      //通过反复执行 foo()函数将数组中元素逐一显示到名为 Result 的<div>中
                var arr=testArr.grep(/3/);                              //测试 grep()方法，该正则表达式匹配含有 3 的字符串
                for(var i=0;i<arr.length;i++)
                {
                        $("Result").innerHTML+="arr["+i+"]:"+arr[i]+"<br>";
                }
                //测试 inject()方法
                var injectArr=testArr.inject("inject()方法执行结果:",foo1);
                $("Result").innerHTML+=injectArr;
                }
</SCRIPT>
```

秘笈心法

心法领悟 303：对 DOM 的支持。

DOM 是 JavaScript 的一项很重要的内容，但并不是所有的浏览器对 DOM 的支持都一样。一般来说，Mozilla 对 DOM 标准支持最好，支持几乎所有的 DOM Level2，以及部分 DOM Level3。在 Mozilla 之后，Opera 和 Safari 也在完成支持上做了突出贡献，极大地缩小了标准之间的差距，支持几乎所有的 DOM Level1 和大部分 DOM Level2。

实例 304 使用 Field 对象操作表单域

光盘位置：光盘\MR\11\304

高级

实用指数：★★★★☆

实例说明

Field 对象用于操作表单元素十分方便。使用该对象的方法，可以很方便地判断某个表单域是否为空、清空表单等。本实例将使用 Field 对象实现操作表单域，运行结果如图 11.8 所示。

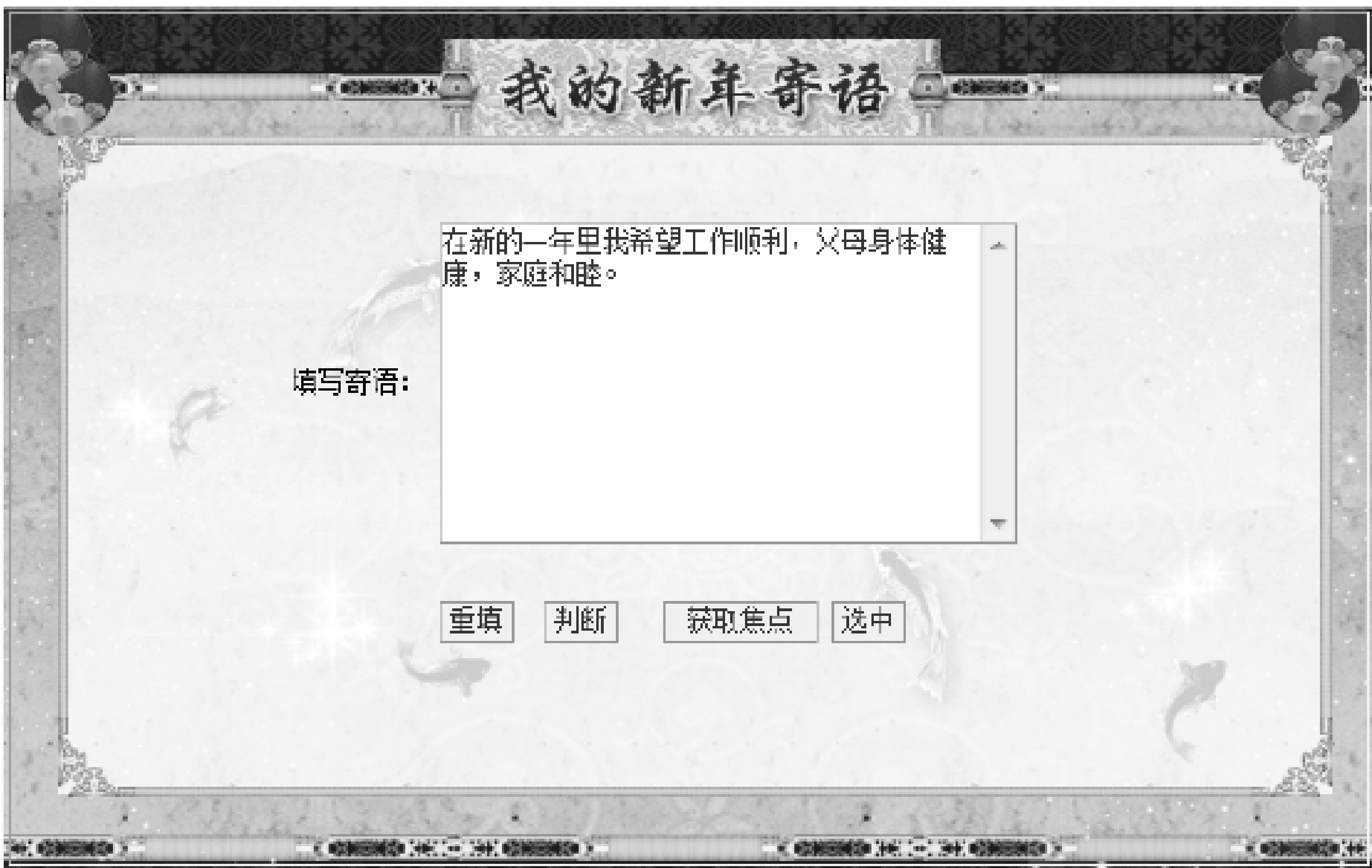


图 11.8　使用 Field 对象操作表单域

关键技术

Field 类定义了操作表单域的相关方法，如表 11.3 所示。

表 11.3　Field 类的常用方法

方　　法	说　　明
clear(field1,field2,…,fieldn)	清除传入该方法的所有表单元素的值。参数 field1～fieldn 既可以是表单元素的 id 属性，也可以是表单元素本身
focus(field)	将焦点移动到指定表单域。该表单域既可以是表单元素的 id 属性，也可以是表单元素本身
select(field)	用于选中表单元素的文本；如果没有要选中的元素，该方法没有任何效果
activate(field)	与 select()方法类似，此方法也可用于选中表单元素的文本，但比 select()多一个功能，如果目标元素没有内容，则将焦点移动到目标元素
present(field1,field2,…,fieldn)	判断参数表单域是否为空，如果不为空则返回 true

（1）在页面中定义表单，并添加文本域与按钮，当用户单击按钮时，会调用相应的 JavaScript 函数。具体代码如下：

```
<table width="379" height="218" border="0" align="center">
  <tr>
    <td width="67" height="141"><div align="right">填写寄语：</div></td>
    <td colspan="4"><div align="left">
      <label>
      <textarea name="textarea" cols="40" rows="10"></textarea>                    //在页面中添加表单域
      </label>
    </div></td>
  </tr>
  <tr>
    <td height="43"> </td>
    <td width="42"><label>
      <input name="Submit" type="button" value="重填"   onclick="clearT()"/>        //单击按钮调用 JavaScript 方法
    </label></td>
    <td width="48"><label>
      <input type="button" name="Submit2" value="判断" onclick="judgeT()"/>
    </label></td>
    <td width="68"><label>
      <input type="button" name="Submit3" value="获取焦点" onclick="getFocusT()" />
    </label></td>
    <td width="120"><input type="button" name="Submit4" value="选中" onclick="selectT()"/></td>
  </tr>
</table>
```

（2）在页面中定义 JavaScript 函数，其中包括清空及选中表单域内容、让表单域获取焦点等方法。具体代码如下：

```
<SCRIPT LANGUAGE="JavaScript">
      function clearT(){
            Field.clear("textarea");                         //清除 id 属性值为 textarea 的表单元素内容
      }
      function judgeT(){
            if(!Field.present("textarea")){                  //判断表单元素是否为空
                  alert("此文本框为空!");
            }
                  else
            {
                  alert("此文本框不为空!");
            }
      }
      function getFocusT(){                                  //让表单元素获取焦点
            Field.focus("textarea");
      }
      function selectT(){                                    //选中表单元素内容
            Field.select("textarea");
      }
</SCRIPT>
```

秘笈心法

心法领悟 304：不同浏览器的显示效果。

使用 CSS 样式或 div 对页面进行设计时，有时使用不同的浏览器显示的效果可能会不同，建议尽量少使用浏览器间显示效果不一样的属性值。

实例 305　通过 Form 对象使表单元素失效

光盘位置：光盘\MR\11\305　　高级　　实用指数：★★★★

Form 对象是 Prototype 中定义的用于操作表单的对象，该对象提供了非常实用的方法，可以获取表单元素的值等。本实例将使用 Form 对象实现对表单的控制，运行结果如图 11.9 所示。

图 11.9　通过 Form 对象使表单元素失效

关键技术

Form 对象提供了一些方法用于操作特定表单的元素，包括访问表单中的表单域，使表单域生效、失效等，如表 11.4 所示。

表 11.4　Form 对象的常用方法

方　　法	说　　明
enable(form)	参数表单的 id 属性，用于使指定表单中所有表单域生效
disable(form)	使指定表单中的所有表单域失效
serialize(form)	返回指定表单内的所有表单域的名和值组成的字符串
getElements(form)	以数组形式返回表单中所有表单域的值
findFirstElement(form)	返回表单中第一个有效的表单域
focusFirstElement(form)	将焦点移动到指定表单中第一个可视的、有效的表单域
reset(form)	重置表单。与调用表单对象的 reset()函数作用相同

设计过程

在页面中定义 JavaScript 函数，分别用于获取表单所有 input 元素、使表单中的所有元素失效、使表单中的所有元素生效、获取表单中的所有表单域值。具体代码如下：

```
<script type="text/javascript">
        function getInput(){                         //获取表单所有 input 元素
              var obj=Form.getInputs("form1");       //获取 form1 表单的所有 input 属性
              for(i=0;i<obj.length;i++)              //循环遍历结果数组
              {
              alert(obj[i].value);                   //显示每个结果
              }
        }
        function setEnable(){                        //使表单中的所有元素生效
              Form.enable("form1");                  //使 form1 表单的所有元素生效
        }
        function setDisable(){                       //使表单的所有元素都失效
              Form.disable("form1");
        }
        function getElements(){                      //获取表单中的所有表单域值
              var obj=Form.getElements("form1");
              for(i=0;i<obj.length;i++){
              alert(obj[i].value);
              }
         }
 </script>
```

心法领悟 305：在 JavaScript 中定义变量。

在 JavaScript 中定义变量不同于在 Java 中定义变量，相对要简单一些，因为 Java 中的变量是有数据类型区分的，例如，int 型变量，char 类型变量等，而在 JavaScript 中所有变量都使用 var 定义就可以了，方便简洁。

实例 306　使用 Form.Element 对象返回特定表单域的值

光盘位置：光盘\MR\11\306

高级

实用指数：★★★★

实例说明

Prototype 中的 Form.Element 对象用于操作表单元素，本实例将应用该对象返回特定表单域的值，运行结果如图 11.10 所示。



图 11.10　使用 Form.Element 对象返回特定表单域的值

Form.Element 对象提供了操作表单域的方法，包括将某个表单域的值转换成查询字符串以及获取指定表单域的值，如表 11.5 所示。

表 11.5　Form.Element 对象的常用方法

方　法	说　明
activate(element)	移动焦点并且选择支持文本选择的表单元素的值。参数 element 表示表单元素对象引用或 id
clear(element)	将表单元素的值清空
disable(element)	禁用表单元素
enable(element)	启用表单元素
focus(element)	将焦点移动到指定表单元素上
getValue(element)	返回指定表单域的值。参数 element 既可以是元素的 id 属性，也可以是元素本身
persent(element)	仅当所有表单元素包含非空值时返回 true，否则返回 false
serialize(element)	将指定元素的名称和值转换为 name=value 的形式

设计过程

（1）在页面中定义 JavaScript 函数，实现获取指定表单域的值。具体代码如下：

```
<script type="text/javascript">
    function getTest(text){                          //定义获取表单域的值
   alert(Form.Element.serialize(text));              //返回指定表单域的查询字符串
}
    function getValue(){                             //定义返回指定表单域的值
   alert(Form.Element.getValue("textfield"));
}
</script>
```

（2）在页面中定义按钮，当单击相应的按钮时，调用相应的 JavaScript 函数。具体代码如下：

```
<tr>
   <td><div align="right"></div></td>
   <td><div align="left">
      <label>
      <input type="button" name="Submit" value="转换查询字符串一" onclick="getTest('textfield')"/>
      </label>
      <label>
      <input type="button" name="Submit2" value="转换查询字符串二" onclick="getTest('textfield2')"/>
      </label>
   </div></td>
</tr>
 <tr>
   <td><div align="right"></div></td>
   <td><div align="left">
      <label>
```

```
        <input type="button" name="Submit" value="返回第一个有效的表单域" onclick="getValue()"/>
        </label>
    </div></td>
</tr>
```

秘笈心法

心法领悟 306：使用重置按钮的注意事项。

重置表单在 Web 开发人员中并不是很受欢迎，因为重置按钮和提交按钮经常会混在一起，很容易错误单击。如果表单是第一次加载，在字段中已包含一些默认信息，那么重置按钮还有些作用，可以恢复到初始值；但如果字段中没有任何初始值，则最好避免使用重置按钮。

11.3　对 Ajax 的支持

实例 307	Ajax.Request 对象发送请求 光盘位置：光盘\MR\11\307	高级 实用指数：★★★

实例说明

前面介绍了 Prototype 对 JavaScript 类库的支持，除此之外，它对 Ajax 也提供了很好的支持。本实例通过 Ajax.Request 类实现发送请求，运行结果如图 11.11 所示。



图 11.11　发送请求



Ajax.Request 类用于创建并处理 Ajax 请求。一个 Ajax.Request 对象有多种用途，如处理请求的生命周期、指定回调函数等。Ajax.Request 类的构造函数语法如下：

```
new Ajax.Request(url,options);
```

参数说明

❶ url：请求的 URL。

❷ options：一个匿名对象，该对象可完成异步请求的发送，通常包含如表 11.6 所示的属性。

表 11.6　option 属性的可选项

选　项	返　回　值	说　明
method	'post'	请求使用的 HTTP 方法，只能为'post'与'get'，注意大小写敏感，不要写成 POST 或者 GET
encoding	'UTF-8'	请求内容的编码
parameters	''	请求的参数。应采用 name=value 的形式

续表

选　　项	返　回　值	说　　明
asynchronous	true	是否异步发送请求，默认值是 true
onComplete	"	用于请求的回调函数

设计过程

（1）在项目的 index.jsp 页面中，定义 JavaScript 函数 searchFruit()，实现监控目标文本框输入文字发生改变的函数。具体代码如下：

```
function searchFruit(){
	var url = 'tips.jsp';                              //定义请求地址
	var params = Form.Element.serialize('favorite');   //将 favorite 表单域的值转换为请求参数
	var myAjax = new Ajax.Request(                     //创建 Ajax 对象
	url,{
		method:'post',                             //设置请求方法
		parameters:params,                         //设置请求参数
		onComplete:showResponse,                   //指定回调函数
		asynchronous:true                          //是否异步发送请求
		});
}
```

（2）定义回调函数 showResponse()。具体代码如下：

```
function showResponse(originalRequest){                          //定义回调函数
	$('result').innerHTML = originalRequest.responseText;    //在提示 div 对象中输出服务器的响应
	Element.show("result");                                  //显示提示 div 对象
}
new Form.Element.Observer("favorite",1,searchFruit);             //为表单域绑定事件处理函数
```

（3）在 tips.jsp 页面中获取请求，并作出相应处理。具体代码如下：

```
<%
	String hdchar = request.getParameter("result");          //获取请求参数
	if("apple".startsWith(hdchar)){                          //判断请求参数是否以 apple 开头
		out.print("苹果");                               //页面输出内容
	}
	else if("banana".startsWith(hdchar)){
		out.print("香蕉");
	}
	else if("peach".startsWith(hdchar)){
		out.println("桃子");
	}
	else if("girl".startsWith(hdchar)){
		out.println("女孩");
	}
	else{
		out.println("没有本单词");
	}
%>
```

秘笈心法

心法领悟 307：Ajax.Request 构造函数的 options 参数。

Ajax.Request 构造函数的 options 参数的 method 属性值不能是 GET 和 POST，只能是 post 和 get，这点一定要注意。

实例 308	注册全局的事件处理器 光盘位置：光盘\MR\11\308	中级 实用指数：★★★★

实例说明

Ajax.Responders 是一个全局监听器的仓库，此对象维护一个基于 Prototype 的 Ajax 相关事件发生时将被调

用的对象列表。对于每个 Ajax 请求，Ajax.Responders 对象都会接收到。本实例应用 Ajax.Responders 注册一个全局的 Ajax 事件处理器，实现在向服务器发送异步请求后，服务器响应没有完成时，页面显示图片；一旦 Ajax 交互完成就会自动隐藏，如图 11.12 所示。



图 11.12 注册全局的事件处理器

关键技术

对于每个 Ajax 请求，Ajax.Responders 对象都会接收到。可以用此对象为 Ajax 操作进行全局日期存储和异常处理等。Ajax.Responders 对象的常用方法如表 11.7 所示。

表 11.7 Ajax.Responders 对象的常用方法

方 法	说 明
register(handler)	注册一个全局 Ajax 事件处理器。该事件处理器应包含名如 Ajax 事件的系列方法（如 onCreate、onComplete、onException）的属性
unregister(handler)	删除一个已经注册的 Ajax 事件处理器
dispatch(callback,request,transport,json)	遍历被注册的处理器列表

设计过程

本实例是在实例 307 的基础上作了修改，使用 Ajax.Responders 注册一个全局的 Ajax 事件处理器，处理器指定了 3 个回调函数，分别在刚开始 Ajax 交互时、Ajax 交互失败时和 Ajax 交互成功时触发。具体代码如下：

```
var myGlobalHandlers = {                                  //定义 Ajax 事件处理器
        onCreate:function()                               //开始 Ajax 交互时触发的方法
        {
                Element.show('Loadimg');
        },
        onFailure:function(){                             //交互失败时触发的方法
                alert('对不起！ \n 页面加载出错！ ');
        },
        onComplete:function(){                            //交互成功时触发的方法
                if(Ajax.activeRequestCount == 0){
                        Element.hide('Loadingimg');
                }
        }

};
Ajax.Responders.register(myGlobalHandlers);
```

心法领悟 308：页面中的错误提示。

在页面中使用 JavaScript 语句时，若发现错误将弹出一个错误的提示框。在此要注意的是，给出的错误行号不一定是正确的，如果产生错误的代码在 HTML 页面中，则该行号可以正确对应发生错误的那一行，但如果发

生错误的代码在外部文件中，则行号往往要差一行。

实例 309	定时刷新时间 光盘位置：光盘\MR\11\309	中级 实用指数：★★★★

实例说明

利用 Ajax.PeriodicalUpdater 类可以周期性地发送 Ajax 请求并根据服务器响应文本更新 HTML 元素的内容。本实例将应用 Ajax.PeriodicalUpdater 实现定时刷新时间，运行结果如图 11.13 所示。

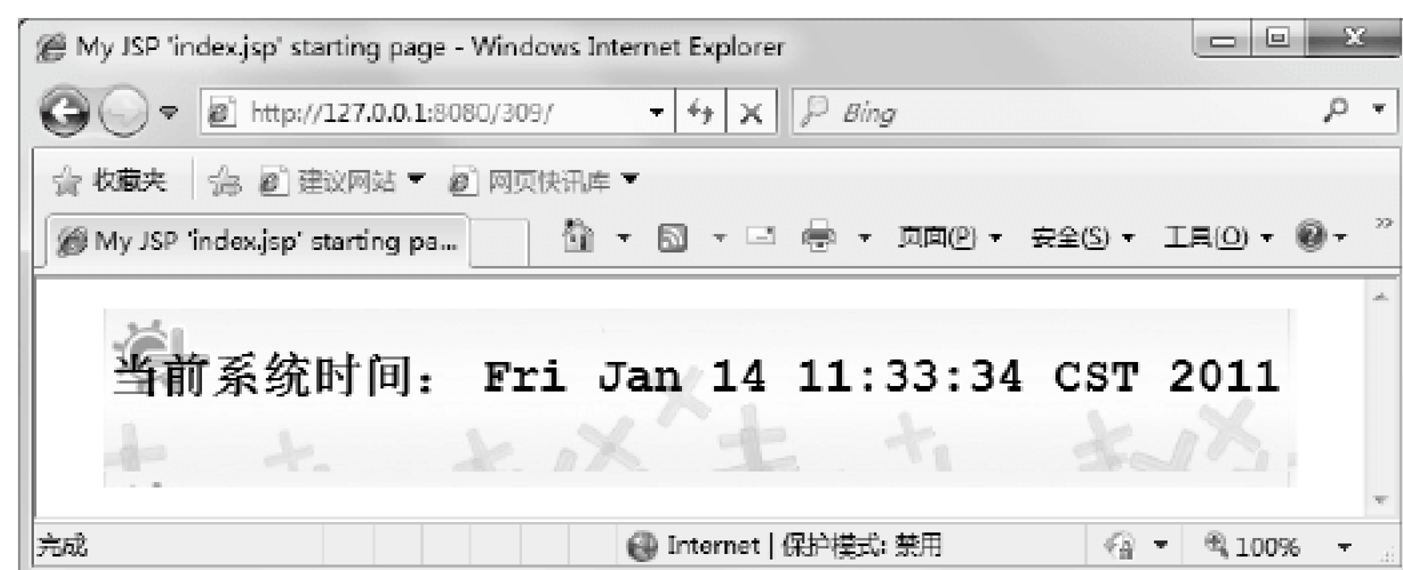


图 11.13　定时刷新时间

本实例应用 Ajax.PeriodicalUpdater 类实现定时刷新。该类的构造方法如下：

```
new Ajax PeriodicalUpdater(container,url[,options]);
```

参数说明

❶ container：要更新的 HTML 元素对象的应用或元素 id。

❷ url：请求的 URL 地址。

❸ options：包含多个选项参数的 Hash 对象。其属性如表 11.8 所示。

表 11.8　options 包含的属性

属　性	说　明
method	该属性指定发送请求的方法，通常使用 post()与 get()
parameters	发送请求的请求参数
frequency	该属性指定多长时间发送一次 Ajax 请求（参数为秒）
decay	如果服务器的两次响应完全相同，则减慢发送请求的频率
onSuccess	当服务器响应成功时，触发该属性指定的函数
onFailure	当服务器响应失败时，触发该属性指定的函数
evalScripts	该属性值只能为 true 和 false，用于指定是否执行服务器响应中的 JavaScript 脚本

设计过程

（1）在 index.jsp 页面中定义表格，实现显示系统时间。具体代码如下：

```
<form id="form1" name="form1" method="post" action="">
  <table width="585" height="87" border="0" align="center" background="images/banner14.jpg">
    <tr>
      <td width="179"> <h2><span class="STYLE1">当前系统时间</span>：</h2></td>
      <td width="396"><div align="left"><h2><span id='time' class="STYLE1"></span></h2></div></td>
    </tr>
  </table>
</form>
```

（2）在 index.jsp 页面中定义 JavaScript 函数，实现定义提交请求。具体代码如下：

```
<script type="text/javascript">
```

```
            var url = 'server.jsp';                          //服务器发送请求的 URL 地址
            var myAjax = new Ajax.PeriodicalUpdater(         //创建 Ajax.PeriodicalUpdater 对象
            'time',url,
            {
                method:'post',
                Parameters:null,
                frequency:1                                  //请求间隔秒数
        }
);
 </script>
```

（3）定义处理请求的 server.jsp 页面，在该页面中实现将当前时间输出。具体代码如下：

```
<body>
    <%
        out.print(new Date());          //在页面中输入当前时间
     %>
</body>
```

秘笈心法

心法领悟 309：Unicode 在 JavaScript 中的表示形式。

所有的 Unicode 字符，包括 ASCII，在 Unicode 中表示为 4 位十六进制数值；前面加上一个“\u”以表示这是一个 Unicode 字符。例如，\u0045 是 E 的 Unicode 形式。

第12章 jQuery框架

- DOM技术
- 表单处理
- 操作表格
- 其他特效
- 对Ajax的支持

12.1 DOM 技术

实例 310 获取文本框中的文本

光盘位置：光盘\MR\12\310

高级

实用指数：★★★★

实例说明

随着 Web 2.0 的兴起，JavaScript 越来越受到重视，一系列 JavaScript 库也蓬勃发展起来。从早期的 Prototype、Dojo 到 jQuery 再到 ExtJS，已经不能让热爱编程的用户对 JavaScript 无动于衷，而 jQuery 以其特有优势，越来越受到编程者的追捧。本实例将应用 jQuery 实现获取文本框中的内容。程序运行初始页时，邮箱地址文本框的初始值为“请输入邮箱地址”，邮箱密码文本框的初始值为“请输入邮箱密码”。当这两个文本框得到焦点时，文本框的初始值会自动消除，要求用户填写内容，本实例的运行结果如图 12.1 所示。



图 12.1 获取文本框中的文本

关键技术

jQuery 是继 Prototype 之后又一个优秀的 JavaScript 类库，由 John Resig 创建于 2006 年 1 月。要在项目中使用 jQuery 类库，首先要配置 jQuery 环境。进入 jQuery 官方网站 http://jquery.com 下载最近版本（本书是以 1.3.2 版本为基础进行讲解的）。同 Prototype 一样，jQuery 也不需要安装，只要将下载的 jquery-1[1].3.2.js 引入到项目中即可。示例代码如下：

```
<script type="text/javascript" src = "JS/jquery-1[1].3.2.js"></script>
```

jQuery 初学者经常分辨不清哪些是 jQuery 对象，哪些是 DOM 对象。每个 DOM（文档类型模型）都可以表示成树形结构。jQuery 对象就是通过 jQuery 包装 DOM 对象后产生的对象。jQuery 对象是 jQuery 独有的。如果一个对象是 jQuery 对象，那么就可以使用 jQuery 中的方法。DOM 对象可以使用 DOM 中的方法，而 jQuery 对象不可以使用 DOM 中的方法，但 jQuery 对象提供了一套更加完善的工具用于操作 DOM。

下面介绍 3 个对于本实例的实现很重要的方法。

- focus()方法：相当于JavaScript中的onfocus()方法，作用是处理获取焦点时的事件。
- blur()方法：相当于JavaScript中的onblur()方法，作用是处理失去焦点时的事件。
- val()方法：该方法不仅能设置元素的值，也可以获取元素的值。

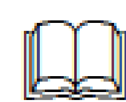
（1）在页面中定义表格，为用户提供可添加的信息。具体代码如下：

```
<table width="701" height="589" border="0" align="center" background="images/dlbg.jpg">
 <tr>
 <td height="117" colspan="2"> </td>
  </tr>
 <tr>
   <td width="554" height="397"><table width="384" height="133" border="0" align="center">
     <tr>
       <td width="180"><label>
         <input name="text" type = "text" id = "address" value="请输入邮箱地址"/>
       </label></td>
       <td width="90"><label>
         <input type="radio" name="radiobutton" value="radiobutton" />
```

```
            留在首页</label></td>
          <td width="90"><p>
              <label>
              <input type="radio" name="radiobutton" value="radiobutton" />
                进入邮箱</label>
              <br />
              <label></label>
              <br />
          </p></td>
        </tr>
        <tr>
          <td><label>
            <input name="text2" type = "text" id = "password" value = "请输入邮箱密码" />
          </label></td>
          <td><label>
            <input type="submit" name="Submit" value="登录" />
          </label></td>
          <td><label>
            <input type="checkbox" name="checkbox" value="checkbox" />
            记住状态</label></td>
        </tr>
        <tr>
          <td>免费注册</td>
          <td>忘记密码？</td>
          <td> </td>
        </tr>
      </table></td>
      <td width="131" rowspan="2"> </td>
    </tr>
    <tr>
      <td> </td>
    </tr>
  </table>
```

（2）在页面中定义 JavaScript 函数，分别在邮箱地址和邮箱密码文本框失去焦点和获取焦点时调用。具体代码如下：

```
<script type="text/javascript">
$(function(){
        $("#address").focus(function(){                    //当 address 获取焦点时调用的函数
            var txt_value = $(this).val();                  //获取文本框值
            if(txt_value == "请输入邮箱地址"){              //判断文本框值
                $(this).val("");                            //将文本框值清空
            }
        });
        $("#address").blur(function(){                     //当 address 失去焦点时调用函数
            var txt_value = $(this).val();
            if(txt_value == ""){                            //如果文本框值为空
                $(this).val("请输入邮箱地址");              //设置文本框值
            }
        });
        $("#password").focus(function(){
            var txt_value = $(this).val();
            if(txt_value == "请输入邮箱密码"){
                $(this).val("");
            }
        });
        $("#password").blur(function(){
            var txt_value = $(this).val();
            if(txt_value == ""){
                $(this).val("请输入邮箱密码");
            }
        });
        });
</script>
```

 **说明：** 在 jQuery 类库中，$就是 jQuery 的一种简写形式，例如，$("#foo")和 jQuery("#foo")是等价的。this 指向当前的文本框，this.defaultValue 就是当前文本框的默认值。

秘笈心法

心法领悟 310：jQuery 库和其他库的冲突。

jQuery 和其他 JavaScript 库（如 Pototype）同时使用时，可能会发生冲突。如果 jQuery 库在其他库之后导入，可以在任何时间调用 jQuery.noConflict()函数，将变量$的控制权移交给其他 JavaScript 库。如果 jQuery 库在其他库之前导入，那么可以直接使用 jQuery 来做一些 jQuery 的工作。这样，可以使用$()方法作为其他库的快捷方式。这里无须使用 jQuery.noConflict()函数。

实例 311　利用 jQuery 实现查找节点

光盘位置：光盘\MR\12\311

高级

实用指数：★★★★

实例说明

利用 jQuery 实现在文档中查找节点是非常容易的，可以使用 jQuery 选择器来完成。本实例通过 jQuery 选择器查找到需要的元素之后，通过 attr()方法获取其属性并将其显示出来，如图 12.2 所示。

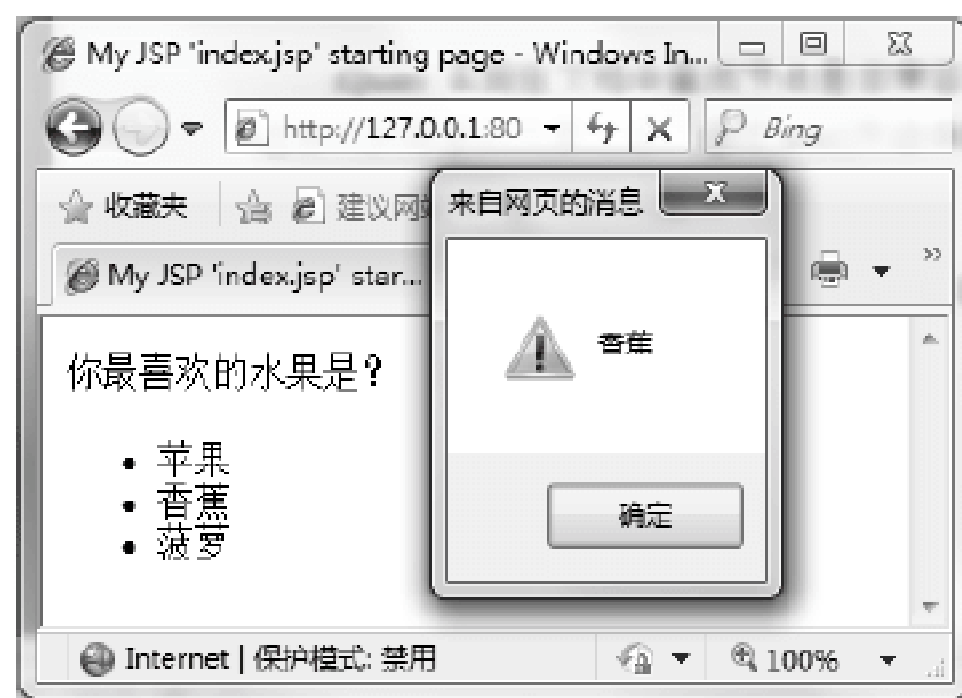


图 12.2　查找节点

本实例的实现使用了 jQuery 选择器，选择器是 jQuery 的根基，在 jQuery 中，对事件处理、遍历 DOM 和 Ajax 操作都依赖于选择器。熟练地使用选择器，可以达到事半功倍的效果。选择器可以分为基本选择器、层次选择器、过滤选择器和表单对象属性过滤选择器（简称表单选择器）。

（1）基本选择器

基本选择器是 jQuery 中最常用的选择器。使用基本选择器可以通过 id、class 元素来查找 DOM 元素。例如，给 id 为 one 的元素设置背景色。代码如下：

```
$("#one").css("backgroud","#sd00da");
```

改变 class 为 mini 的所有元素的背景色。代码如下：

```
$(".mini").css("backgroud","#sd00da");
```

（2）层次选择器

层次选择器可通过 DOM 元素之间的层次关系来获取特定元素，如后代元素、子元素、相邻元素和兄弟元素。例如，获取<div>下名为<span>的子元素。代码如下：

```
$("div>span")
```

获取<div>中的所有<span>元素。代码如下：

```
$("div span")
```

（3）过滤选择器

过滤选择器主要是通过特定的过滤规则来筛选出所需的 DOM 元素，过滤规则与 CSS 中的伪类选择器语法相同，选择器以冒号（:）开头。例如，选取<div>元素中的第一个<div>元素。代码如下：

```
$("div:first")
```

选取<div>元素中的最后一个<div>元素。代码如下：

```
$("div:last")
```

（4）表单对象属性过滤选择器

表单对象属性过滤选择器主要是对所选择的表单元素进行过滤。多用于下拉表框、单选按钮、复选框等。例如，获取所有可用元素。代码如下：

```
$("#form1:enabled");
```

选取所有被选中的单选按钮、复选框。代码如下：

```
$("input : checked");
```

获取所有被选中的下拉列表选项元素。代码如下：

```
$("select:selected");
```

设计过程

（1）设计页面，在其中显示“最喜欢的水果”。具体代码如下：

```
<body>
        <p title="选择你最喜欢的水果">你最喜欢的水果是？</p>
          <ul>
          <li title = '苹果'>苹果</li>
          <li title = '香蕉'>香蕉</li>
          <li title = '菠萝'>菠萝</li>
          </ul>
</body>
```

（2）在页面中定义 jQuery 函数，实现获取页面元素并显示。具体代码如下：

```
<script type="text/javascript">
        $(function () {
                var $li = $("ul li:eq(1)");            //获取索引为 1 的 li 对象
                var li_txt = $li.text();               //获取对象属性值
                alert(li_txt);                         //显示对象属性值
        });
</script>
```

秘笈心法

心法领悟 311：使用选择器对表单进行统计。

使用选择器可以实现对表单元素的统计。例如，想得到表单内表单元素的个数。代码如下：

```
$("#form1:input").length;
```

如果想得到表单内单行文本框的个数。代码如下：

```
$("#form1:text").length;
```

如果想得到表单内密码框的个数。代码如下：

```
$("#form1:password".length);
```

实例 312 动态为表格追加样式

光盘位置：光盘\MR\12\312

高级

实用指数：★★★★

jQuery 框架提供了很多方法用于在页面中添加相应的样式。本实例实现为页面中的表格添加 CSS 样式，页面的初始效果如图 12.3 所示，添加样式后的效果如图 12.4 所示。

用户名	地域	订单
小陈	长春	100000
小李	沈阳	21546
小葛	北京	659810
	为表格添加样式	

图 12.3 初始页面

用户名	地域	订单
小陈	长春	100000
小李	沈阳	21546
小葛	北京	659810
	为表格添加样式	

图 12.4 添加样式后的效果

关键技术

本实例实现在页面中追加样式，使用的是 addClass()函数。此外，jQuery 还提供了一个专门用来设置 CSS 样式的方法 attr()。两者的区别在于，如果原页面中包含有样式，使用 attr()方法会将原样式取消，而使用 addClass()方法则会保存原有的样式。下面举例说明其区别，如表 12.1 所示。

表 12.1 attr()和 addClass()的区别

实 例 说 明	addClass()	attr()
第 1 次使用	$("p").addClass("high")	$("p").attr("class","high")
第 1 次结果	<p class ="high">test</p>	<p class ="high">test</p>
第 2 次使用	$("p").addClass("another");	$("p").attr("class","another");
最终结果	<p class ="high another">test</p>	<p class ="another">test</p>

addClass()函数的语法如下：

```
addClass(Class)
```

参数说明

Class：要追加的 CSS 样式。

设计过程

（1）在页面中定义 CSS 样式，在该样式中设置背景色、字体样式等。具体代码如下：

```
<style type="text/css">
        .sd{
                font-weight:bold;
                color:black;
                background: red;
                }
        </style>
```

（2）在页面中定义 JavaScript 代码，实现为页面追加 CSS 样式。具体代码如下：

```
<script type="text/javascript">
        function clicks(){
                $("#table").addClass("sd");
                }
</script>
```

说明：上段代码中的 table 表示的是页面中表格的 id 属性。

（3）在页面中添加表格对象，具体代码参见配书光盘中的源程序。

秘笈心法

心法领悟 312：jQuery 无法使用 DOM 对象的任何方法。

注意，类似$("#id").innerHTML 和$("#id").checked 的写法都是错误的，可以用$("#id").html()和$("#id").attr("checked")等 jQuery 方法来代替。同理，DOM 对象也不能使用 jQuery 中的方法。在 jQuery 中使用 document.getElement ById("id").html()也会报错。

实例 313 动态为表格移除样式 中级

光盘位置：光盘\MR\12\313 实用指数：★★★★☆

实例 312 演示了如何为表格追加样式，本实例在此基础上稍加改动，添加移除样式的功能，运行结果如图 12.5 所示。

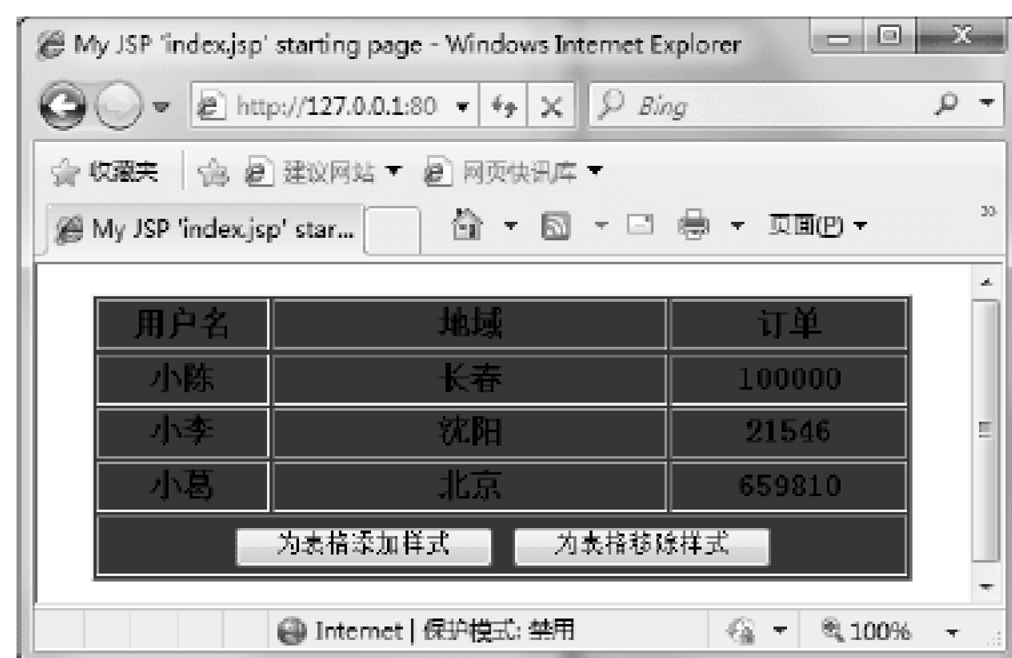


图 12.5　动态为表格移除样式

关键技术

在页面中移除样式的方法为 removeClass()。语法如下：

```
$("p").removeClass("high")
```

参数说明

high：表示要移除的 CSS 样式。

例如，将<p>元素中的两个 class 都删除，可以使用两次 removeClass()方法。代码如下：

```
$("p").removeClass("high").removeClass("as");
```

上述代码也可以写成：

```
$("p").removeClass("high as")
```

如果要将元素<p>中的全部 CSS 样式删除，可以使用不带参数的 removeClass()方法。代码如下：

```
$("p").removeClass();
```

（1）在页面中定义 CSS 样式，设置背景色与字体颜色等。具体代码如下：

```
<style type="text/css">
    .sd{
            font-weight:bold;
            color:black;
            background: red;
        }
</style>
```

（2）在页面中定义 JavaScript 函数，实现为表格添加样式与移除样式。具体代码如下：

```
<script type="text/javascript">
        function clicks(){
                $("#table").addClass("sd");                    //定义向表格添加样式函数
            }
        function removeClick(){
            $("#table").removeClass("sd");                     //定义从表格移除样式函数
            }
</script>
```

（3）在页面中定义表格，在表格中添加按钮，单击按钮调用不用的方法，实现为表格添加样式和移除样式。具体代码如下：

```
<form id="form1" name="form1" method="post" action="">
 <table width="428" height="148" border="1" align="center" id="table">
   <tr>
     <td width="86"><div align="center">用户名</div></td>
     <td width="201"><div align="center">地域</div></td>
     <td width="119"><div align="center">订单</div></td>
   </tr>
   <tr>
     <td><div align="center">小陈</div></td>
     <td><div align="center">长春</div></td>
     <td><div align="center">100000</div></td>
   </tr>
   <tr>
     <td><div align="center">小李</div></td>
     <td><div align="center">沈阳</div></td>
     <td><div align="center">21546</div></td>
```

```
    </tr>
<tr>
        <td><div align="center">小葛</div></td>
        <td><div align="center">北京</div></td>
        <td><div align="center">659810</div></td>
    </tr>
    <tr>
        <td colspan="3"><div align="center">
            <label>
            <input type="button" name="Submit" value="为表格添加样式"   onclick="clicks()"/>
            </label>
            <label>
            <input type="submit" name="Submit2" value="为表格移除样式" onclick="removeClick()" />
              </label>
          </div>          <div align="center"></div>          <div align="center"></div></td>
      </tr>
    </table>
</form>
```

秘笈心法

心法领悟 313：#id 选择符。

用#id 作为选择符获取的是 jQuery 对象，而并非 document.getElementById("id")所得到的 DOM 对象，两者并不等价。从学习 jQuery 开始就应当树立正确的观念，分清 jQuery 对象和 DOM 对象之间的区别，保证程序的正确性。

实例 314　实现表格的样式切换

光盘位置：光盘\MR\12\314

高级

实用指数：★★★★

实例说明

jQuery 提供了 toggleClass()方法用于控制样式的重复切换。如果类名存在则将其删除，如果类名不存在则进行添加。本实例将在页面中添加“为表格切换样式”按钮，单击该按钮可实现指定的样式切换，如图 12.6 所示。

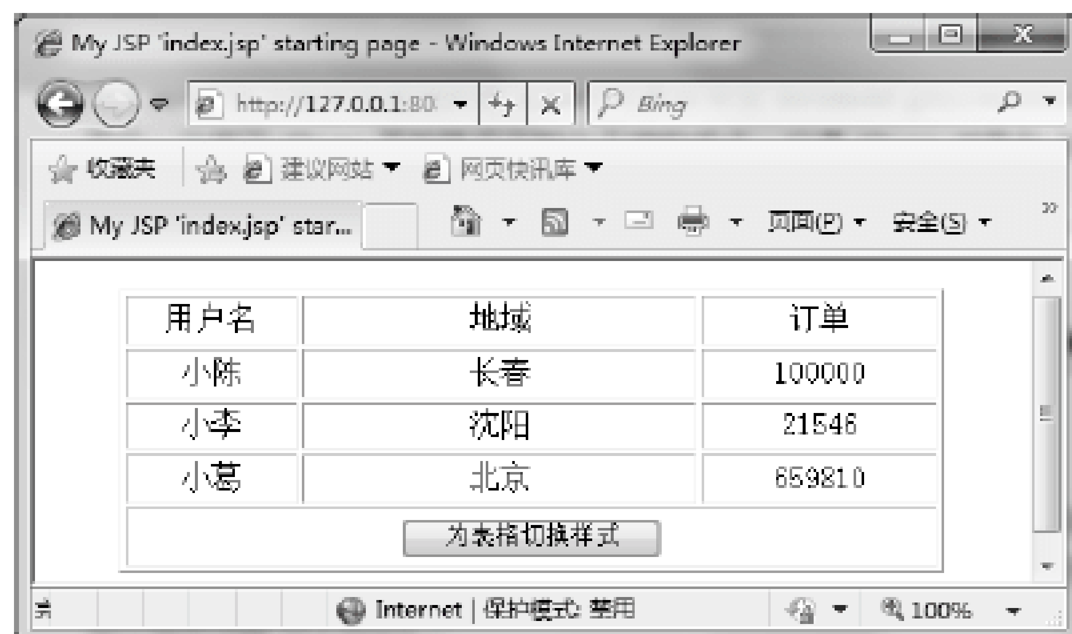


图 12.6　实现表格的样式切换

本实例使用 jQuery 的 toggleClass()方法控制样式的重复切换。语法如下。

```
$("p").toggleClass("another")
```

参数说明

another：要切换的 CSS 样式。

例如，<p>元素的原始代码如下：

```
<p>你最喜欢的水果是？</p>
```

实现切换 CSS 样式后，代码如下：

```
<p class ="another">你最喜欢的水果是？</p>
```

再次单击“切换样式”按钮后，<p>元素又返回原来的状态。代码如下：

```
<p>你最喜欢的水果是？</p>
```

设计过程

（1）在页面中定义 CSS 样式，设置背景色和字体颜色。具体代码如下：

```
<style type="text/css">
    .sd{
            font-weight:bold;
            color:black;
            background: red;
        }
</style>
```

（2）在页面中定义 JavaScript 函数，实现为页面切换 CSS 样式。具体代码如下：

```
<script type="text/javascript">
        function clicks(){
                $("#table").toggleClass("sd");                    //定义为表格切换样式函数
        }
</script>
```

秘笈心法

心法领悟 314：在 Dreamweaver 中编写 jQuery 代码。

Dreamweaver 是建立 Web 站点和应用程序的专业工具。要使 Dreamweaver 支持 jQuery 自动提示代码功能，方法非常简单，只需要下载一个名为 jQuery_API.mxp 的插件即可。在 Dreamweaver 中依次选择“命令”/“扩展管理”/“安装扩展”/jQuery_API.mxp 命令后，即可自动安装插件。

12.2　表 单 处 理

实例 315　实现表单文本域的放大和缩小　中级
光盘位置：光盘\MR\12\315　实用指数：★★★★

实例说明

在浏览网站时，经常可以看到在一些评论网站中有“放大”和“缩小”等按钮，用来控制文本域的大小。在 jQuery 框架中实现这样的功能很简单。本实例将应用 jQuery 框架，实现将留言文本域放大和缩小，运行结果如图 12.7 所示。



图 12.7　实现表单文本域的放大和缩小

关键技术

jQuery 定义了 4 种方法来实现元素的宽和高的控制，分别介绍如下。

- height()：获取元素的高度值，单位为像素。
- width()：获取元素的宽度值，单位为像素。
- height(val)：为每个匹配的元素设置 CSS 高度属性值，如果没有明确指定单位，默认使用像素。
- width(val)：为每个匹配的元素设置 CSS 宽度属性值。如果没有明确指定单位，默认使用像素。

（1）在页面中定义表格，在其中定义“放大”和“缩小”两个按钮，实现页面布局。具体代码如下：

```
<tr>
    <td height="46"><div align="right"></div></td>
```

```
    <td><div align="left">
      <label>
      <input type="button" name="Submit" value="放大" id = "bigger" />
      </label>
      <label>
      <input type="button" name="Submit2" value="缩小" id = "smaller"   />
      </label>
    </div></td>
  </tr>
```

（2）在页面中定义 JavaScript 函数，实现将文本域进行放大和缩小。具体代码如下：

```
<script type="text/javascript">
   $(function(){
            var $content = $("#content");                          //定义文本域变量
            $("#bigger").click(function(){                         //判断是否单击了“放大”按钮
                  if($content.height()<500){                       //判断高度是否小于 500
                        $content.height($content.height()+50);     //高度加 50 像素
                        }
                  else{
                        alert("已经够高了！！");
                        }
                  });
            $("#smaller").click(function(){                        //判断用户是否单击了“缩小”按钮
                        if($content.height() > 50){                //判断高度是否大于 50 像素
                        $content.height($content.height()-50);     //高度减 50
                        }
                        else{
                        alert("不能再缩小了！！！");
                        }
            });
       });
</script>
```

秘笈心法

心法领悟 315：没有找到相应对象异常。

笔者在编写本实例时，出现了没有找到相应对象的错误提示。仔细检查了程序中的代码，没有发现逻辑错误。后来查看页面的开端才发现，原来是没有导入 jQuery 类库才导致了此问题。因此，在这里提醒读者，如果也遇到了相应问题，应检查是否导入了 jQuery 类库，不要因为一时疏忽而导致不必要的麻烦。

实例 316　实现复选框的全选与反选

光盘位置：光盘\MR\12\316

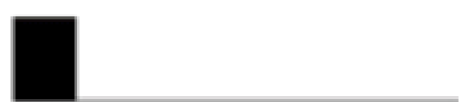
高级

实用指数：★★★★

实例说明

对复选框最基本的应用，就是对复选框的全选与全不选操作。本实例在用户注册页面中添加了多个爱好复选框，并添加了“全选”和“全不选”按钮，用于实现复选框的全选和全不选，如图 12.8 所示。

图 12.8　实现复选框的全选与全不选



实现复选框的全选与全不选，首先需要了解如何让复选框处于选中状态。这可通过复选框元素的 checked 属性来判断，如果 checked 属性的值为 true，说明被选中；如果值为 false，说明未被选中。在 jQuery 框架中可以通过 attr()方法设置 checked 的值，使之被选中。

设计过程

（1）在页面中定义表格，来为用户提供添加的注册信息，其中包含由复选框组成的“爱好”列表，所有复选框的 name 属性全部为 checkbox。具体代码如下：

```
<td><div align="left">
        <p>
        <input type="checkbox" name="checkbox" value="checkbox">
      上网
        <input type="checkbox" name="checkbox" value="checkbox">
        旅游
        <input type="checkbox" name="checkbox" value="checkbox">
        交友
        <input type="checkbox" name="checkbox" value="checkbox">
        逛街                </p>
        <p>
          <input type="checkbox" name="checkbox" value="checkbox">
          看书
          <input type="checkbox" name="checkbox" value="checkbox">
          书法
          <input type="checkbox" name="checkbox" value="checkbox">
          游戏
          <input type="checkbox" name="checkbox" value="checkbox">
           球类</p>
        <p align="right">
          <input type="button" name="Submit" id="checkAll" value="全选">
          <input type="button" name="Submit2" id="checkNo" value="全不选">
        </p>
  </div></td>
```

（2）在页面中定义 JavaScript 函数，来判断用户是否单击了“全选”与“全不选”按钮，并给出相应的操作。具体代码如下：

```
<script type="text/javascript">
      $(function(){
          $("#checkAll").click(function(){                    //判断用户是否单击了“全选”按钮
              $('[name = checkbox]:checkbox').attr('checked',true);     //将复选框设置为全部选中状态
          });
          $("#checkNo").click(function(){                     //判断用户是否单击了“全不选”按钮
              $('[name = checkbox]:checkbox').attr('checked',false);    //将复选框设置为全不选中状态
          });
      });
</script>
```

秘笈心法

心法领悟 316：为什么要将所有复选框的 name 属性设置为相同的值。

本实例将所有复选框的 name 属性设置为 checkbox，这也是实现对复选框操作的常用方法，进行这样的设置后，可以很方便地对复选框进行全选和全不选操作，而无须通过 name 属性逐一进行处理。

实例 317	列表框的综合应用 光盘位置：光盘\MR\12\317	高级 实用指数：★★★★

实例说明

列表框的作用非常多，其中实现两个列表框的值之间的互换比较常见。本实例将实现学生选课信息的添加。运行程序，可以看到在如图 12.9 所示页面中提供了两个列表框，通过单击中间的“>|”、“>>”、“<<”和“|<”按钮，可实现列表框中内容的互换。

图 12.9　列表框的综合应用

■ 关键技术

本实例实现了两项内容，分别为将选中的列表项添加给另一个列表框、将全部的列表项添加给对方。实现这一功能，首先要获取列表框中被选中的选项，然后将其从当前列表框中删除，最后将删除的选项添加给另一个列表框。在 jQuery 框架中使用 appendTo()方法可以直接完成。该方法的具体语法如下：

```
$option = $('#select2');
```

语法说明：将$option 从原来所在的列表框中删除，添加到 select2 中。

■

（1）在页面中定义表格，实现为用户提供可添加的选课信息，其中定义两个列表框。具体代码如下：

```
<td width="125"><div align="center">
  <label>
  <select   multiple="multiple" size="6" name="select" id ="select">
    <option value="英语听说">英语听说</option>
    <option value="声乐">声乐</option>
    <option value="美术">美术</option>
    <option value="书法">书法</option>
    <option value="高数">高数</option>
    <option value="儿童文学">儿童文学</option>
  </select>
  </label>
</div></td>
<td width="51"><label>
  <div align="center">
    <input type="button" name="right" id="right" value="&gt;|" />
    <br />
    <input type="button" name="rightAll" id="rightAll" value="&gt;&gt;" />
    <br />
    <input type="button" name="left" id="leftAll" value="&lt;&lt;" />
    <br />
    <input type="button" name="leftAll" id="left" value="|&lt;" />
  </div>
</label></td>
<td width="117"><div align="center">
  <select   multiple="multiple" size="6" name="select2" id ="select2">
  </select>
</div></td>
```

（2）在该页面中定义 JavaScript 函数，实现通过 jQuery 框架将列表框中的值进行移动。具体代码如下：

```
<script type="text/javascript">
        $(function(){
                $("#right").click(function(){                              //判断用户是否单击了“>>”按钮
                        var $options = $('#select option:selected');      //获取用户选择的列表项
                        var $remove = $options.remove();                   //将列表项删除
                        $remove.appendTo('#select2');                      //将删除的列表项追加到 select2 中
                });
                $('#rightAll').click(function(){                           //判断用户是否单击了“>|”按钮
                        var $options = $('#select option');                //获取全部的列表项
                        $options.appendTo('#select2');                     //将列表项全部添加到 select2 中
                });
```

```
            $("#left").click(function(){                    //判断用户是否单击了“<<”按钮
                var $options = $('#select2 option:selected');
                var $remove = $options.remove();
                $remove.appendTo('#select');
            });
            $("#leftAll").click(function(){
                var $options = $('#select2 option');
                var $remove = $options.remove();
                $remove.appendTo('#select');
            });
        });
</script>
```

秘笈心法

心法领悟 317：为表单定义 id 属性。

jQuery 的某些函数通过表单的 name 属性可以进行绑定，但有些程序或者浏览器不支持使用 name 属性进行绑定，因此为了避免程序出现问题，应尽量在每个表单元素中都添加 id 属性。

实例 318　实现表单验证

光盘位置：光盘\MR\12\318

高级

实用指数：★★★★

实例说明

对表单的验证经常被用在一些类似于注册的页面中。本实例将通过 jQuery 框架实现对表单的验证，其中包括是否输入合法的用户名、是否输入合法的邮箱地址等，如图 12.10 所示。

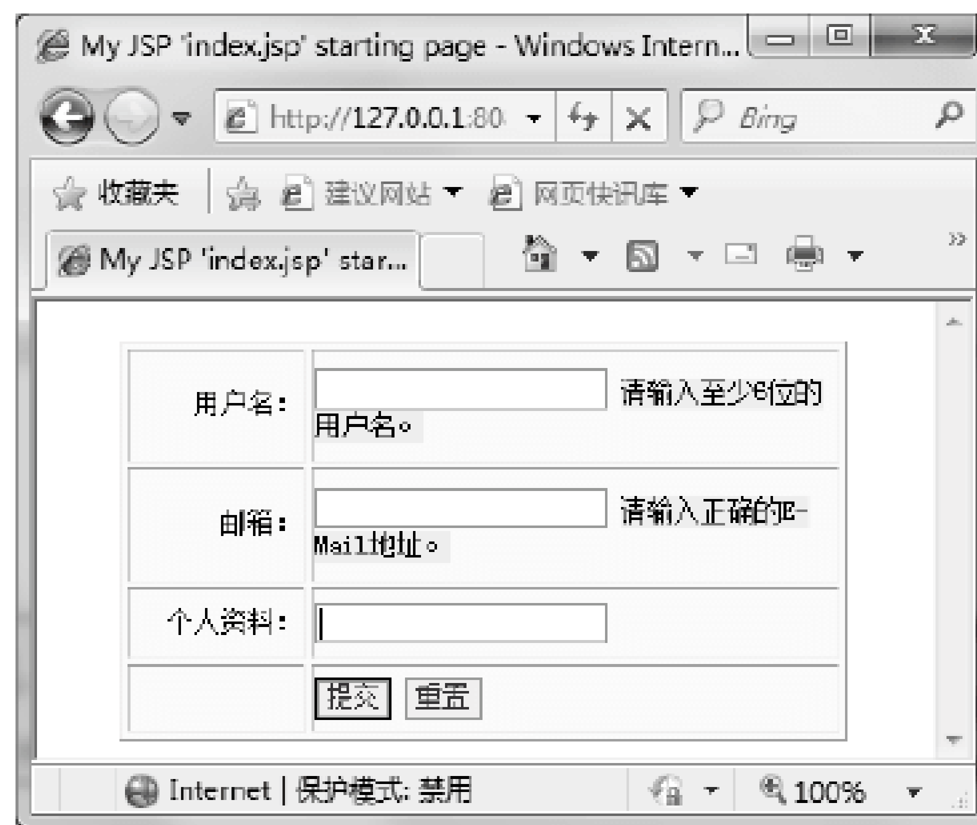


图 12.10　实现表单验证

关键技术

本实例要实现的是对用户名和邮箱地址进行验证，即当用户名与邮箱地址文本框失去焦点时，对用户输入的用户名与邮箱地址进行验证。在此要注意的是，为文本框绑定失去焦点事件，使用的是 blur() 函数。

在页面中定义 JavaScript 函数，实现当文本框失去焦点时触发，并验证用户名和邮箱两个文本框是否符合要求，如果不符合则给出相应的判断。具体代码如下：

```
<script type="text/javascript">
$(function(){
        $(':input').blur(function(){
            var $parent = $(this).parent();
            $parent.find("#s").remove();
            if($(this).is('#username')){
                if(this.value == "" || this.value.lenth < 6){
                    var errorMsg = '请输入至少 6 位的用户名。';
                    $parent.append('<span id ="s" class = "formtips onError">'+errorMsg+'</span>');

                }
                else{
                    var okMsg = '输入正确.';
                    $parent.append('<span id = "s" class = "formtips onSucess">' + okMsg +'</span>');

                }
            }
            if($(this).is('#email')){
                if(this.value == "" || (this.value != "" && !/.+@.+\.[a-zA-Z]{2,4}$/.test(this.value))){
                    var errorMsg = "请输入正确的 E-Mail 地址。";
```

```
                    $parent.append('<span id = "s" class = "formtips onError">' +errorMsg +'</span>');
                }
                else{
                    var okMsg = '输入正确.';
                    $parent.append('<span id = "s" class = "formtips onSuccess">' + okMsg +'</span>');
                }
            }
        });
    });
</script>
```

秘笈心法

心法领悟 318：应用正则表达式匹配超链接地址。

本实例采用正则表达式匹配用户的邮箱地址。应用正则表达式进行验证是很常见的一种形式，例如，使用正则表达式验证超链接地址，代码为“((http|https|ftp):(\\/\\/|\\\\)((\\w)+\\.)+(net|com|cn|org|[0-9]{1,3})"(((\\/[\\/]*|\\[\\/])* (\\w)+)|[.](\\w)+)*(((([?](\\w)+){1}[=]*))*((\\w)+){1}([\\&](\\w)+ [\\=](\\w)+)*)*)”。

实例 319　密码强度检测

光盘位置：光盘\MR\12\319

高级

实用指数：★★★

实例说明

现在各大网站在会员注册时，基本上都提供了密码强度检测的功能。可能读者对此感到很好奇，又不知道如何实现。通过本实例的学习，可以轻松地学会如何使用 jQuery 插件实现密码强度检测的功能。本实例运行结果如图 12.11 所示。



图 12.11　密码强度检测

关键技术

本实例是通过 jQuery 插件 password-strength 实现的。读者可以打开网站 http://simplythebest.net/scripts/ajax/ajax_password_strength.html 或者直接在百度、谷歌搜索，下载 password-strength 插件和 jQuery 框架。解压缩后，将 digitalspaghetti.password.js 和 jquery.1.3.2.min.js 复制到项目根目录的 JS 文件夹下，然后在 Web 页中引入这些文件，再初始化插件 password-strength。代码如下：

```
<script type="text/javascript">
$(function() {
        $('#user_password').pstrength();
});
</script>
```

（1）引入 jQuery 框架、jQuery 插件 password-strength 和相应的样式。代码如下：

```
<script type="text/javascript" src="js/jquery-1.3.2.js"></script>
<script type="text/javascript" src="js/digitalspaghetti.password.js"></script>
```

（2）初始化 jQuery 插件 password-strength，使其能够进行密码强度检测。代码如下：

```
<script type="text/javascript">
$(function() {
        $('#user_password').pstrength();
});
</script>
```

（3）在<body></body>区域中加入 HTML 密码框。运行程序，在文本框中输入密码时，便可以对其强度进行检测。代码如下：

```
<input type="password" id="user_password" name="user_password" />
```

秘笈心法

心法领悟 319：网站中播放 Flash。

Flash 是一种交互式矢量多媒体技术，其前身是 Futureplash——早期网上流行的矢量动画插件。现在网上已经有成千上万个 Flash 站点，比较知名的有 Macromedia 公司的 Shockwave 站点，全部采用了 Shockwave Flash 和 Director。可以说 Flash 已经逐渐成为交互式矢量的标准，并以其美观、实用而深受广大网站开发者的青睐。它可以包含动画、声音和超文本链接等，而且文件体积小，如果将其嵌入到网页中，可以使网页增色不少。

实例 320	文本框提示标签 光盘位置：光盘\MR\12\320	中级 实用指数：★★★★

实例说明

在平时浏览网站时，经常能看到注册会员时，文本框后边有一个小“?”号，当鼠标指针移至“?”时，会弹出一个小标签，标签中有对文本框中数据的一些要求。例如，密码文本框弹出的标签中，要求密码长度为 6～20 位，不能包含特殊字符等。本实例将使用 jQuery 插件实现这一功能，运行结果如图 12.12 所示。

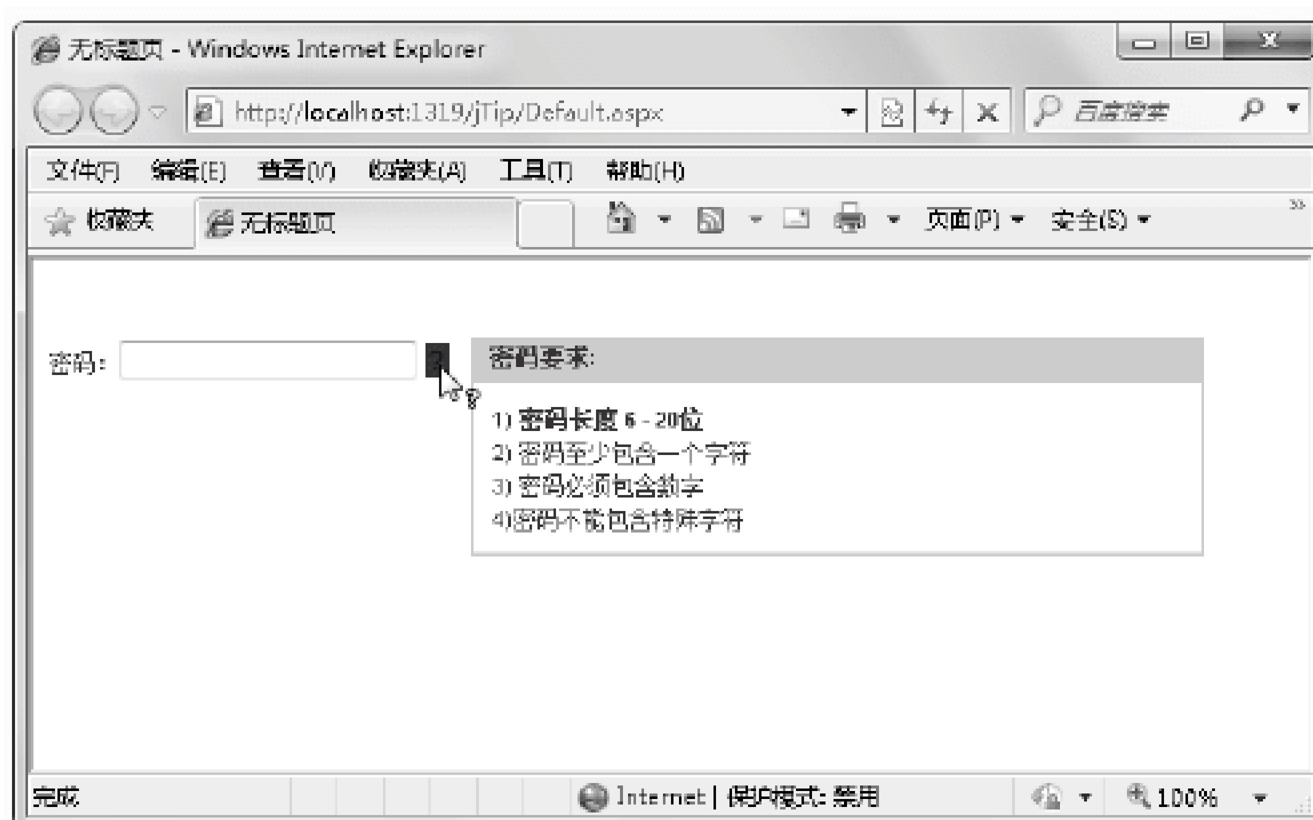


图 12.12　文本框提示标签

关键技术

本实例是通过 jQuery 插件 jTip 实现的，读者可以打开网站 http://www.codylindley.com/blogstuff/js/jtip/jTip.zip 或者直接在百度、谷歌搜索下载。解压缩后，将 JS 和 CSS 文件夹复制到项目根目录下。然后在 Web 页中引入这些文件，以便能够使提示标签生效。代码如下：

```
<script src="js/jquery.js" type="text/javascript"></script>
<script src="js/jtip.js" type="text/javascript"></script>
<link href="css/global.css" rel="stylesheet" type="text/css" />
```

设计过程

（1）在项目的 index.jsp 页面中添加一个 HTML 文本框，用于输入密码。具体代码参见配书光盘中的源程序。

（2）引入 jQuery 插件 jTip 和相应的样式。代码如下：

```
<script src="js/jquery.js" type="text/javascript"></script>
<script src="js/jtip.js" type="text/javascript"></script>
<link href="css/global.css" rel="stylesheet" type="text/css" />
```

（3）在文本框后边加入一个标签，并在该标签中加入一个超链接，其地址为 read1.htm?width=375。其中，read1.htm 是新创建的静态页，在该静态页中输入文本框提示标签显示的内容。代码如下：

```
<span class="formInfo"><a href="read1.htm?width=375" class="jTip" id="one" name="密码要求:">?</a></span>
```

（4）创建一个静态页，命名为 read1.htm，在该静态页中输出文本框提示标签要显示的内容。代码如下：

```
<ul>
<li>1) <strong>密码长度 6 - 20 位</strong></li>
<li>2)密码至少包含一个字符</li>
<li>3)密码必须包含数字</li>
<li>4)密码不能包含特殊字符</li>
</ul>
```

心法领悟 320：jQuery 定义的 4 个嵌套方法。

- wrap(html)方法：把所有匹配的元素分别用指定结构化标签包裹起来。

- warp(element)方法：把所有匹配的元素分别用指定元素包裹起来。
- wrapALl(html)方法：把所有匹配的元素用一个元素包裹起来。
- warpInner(html)方法：把每个匹配的元素的子内容（包括文本节点）使用一个 HTML 结果包裹起来。

12.3 操 作 表 格

实例 321 表格隔行变色

光盘位置：光盘\MR\12\321

中级

实用指数：★★★★

实例说明

在页面设计中，实现表格的隔行变色是很常见的一种功能。隔行变色，就是指表格中奇数行是一种颜色，偶数行是另一种颜色。本实例使用 jQuery 实现表格隔行变色，运行结果如图 12.13 所示。

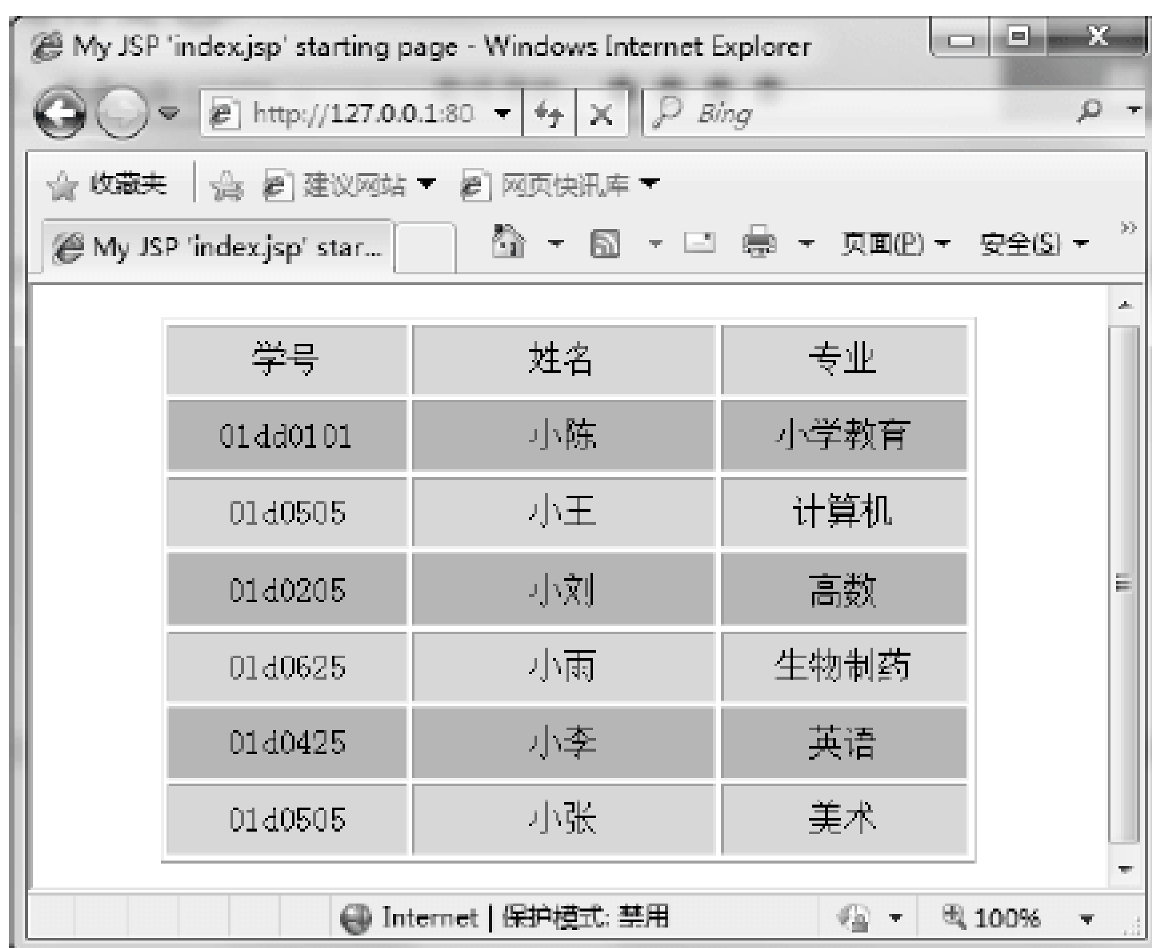


图 12.13　表格隔行变色

关键技术

要实现本实例，首先要求定义 CSS 样式，即表格偶数行与奇数行的样式，之后通过 jQuery 框架的 addClass() 方法进行加载即可。该方法的语法如下：

```
addClass(className)
```

参数说明

className：指定的 CSS 样式。

注意：$("tr:odd")和$("tr:even")选择器中索引是从 0 开始的，因此第一行是偶数。

（1）定义 CSS 样式，包括表格的奇数行样式与偶数行样式。具体代码如下：

```
<style type="text/css">
        .even{                                              //定义偶数行样式
                background: #FFCCCC;
        }
        .odd{                                               //定义奇数行样式
                background: #66CCCC;
        }
</style>
```

（2）定义 JavaScript 函数，实现为表格的偶数行和奇数行分别添加 CSS 样式，进而实现表格的隔行变色。具体代码如下：

```
<script type="text/javascript">
	$(function() {
		$("tr:odd").addClass("odd");		//为奇数行添加样式
		$("tr:even").addClass("even");		//为偶数行添加样式
	});
</script>
```

（3）在页面中定义表格，在其中除了文本内容外，不需要添加任何内容。具体代码如下：

```
<table width="383" height="257" border="1" align="center">
  <tr>
    <td width="121"><div align="center">学号</div></td>
    <td width="171"><div align="center">姓名</div></td>
    <td width="139"><div align="center">专业</div></td>
  </tr>
  <tr>
    <td><div align="center">01dd0101</div></td>
    <td><div align="center">小陈</div></td>
    <td><div align="center">小学教育</div></td>
  </tr>
  <tr>
    <td><div align="center">01d0505</div></td>
    <td><div align="center">小王</div></td>
    <td><div align="center">计算机</div></td>
  </tr>
  <tr>
    <td><div align="center">01d0205</div></td>
    <td><div align="center">小刘</div></td>
    <td><div align="center">高数</div></td>
  </tr>
  <tr>
    <td><div align="center">01d0625</div></td>
    <td><div align="center">小雨</div></td>
    <td><div align="center">生物制药</div></td>
  </tr>
  <tr>
    <td><div align="center">01d0425</div></td>
    <td><div align="center">小李</div></td>
    <td><div align="center">英语</div></td>
  </tr>
  <tr>
    <td><div align="center">01d0505</div></td>
    <td><div align="center">小张</div></td>
    <td><div align="center">美术</div></td>
  </tr>
</table>
```

秘笈心法

心法领悟 321：将表格的某一行变为高亮显示状态。

本实例使用 jQuery 的选择器实现了表格的隔行变色功能，如果希望将表格的某一特定行变为高亮显示状态，则可以使用 contains 选择器。例如：

```
$("tr:contains ").addClass("odd");
```

实例 322	通过单选按钮控制表格的行高亮显示 光盘位置：光盘\MR\12\322	高级 实用指数：★★★★

实例说明

本实例在实例 321 的基础上做了修改，使用单选按钮控制表格中行的高亮显示。在表格中，选中某一行前面的单选按钮，即可实现将该行高亮显示，如图 12.14 所示。

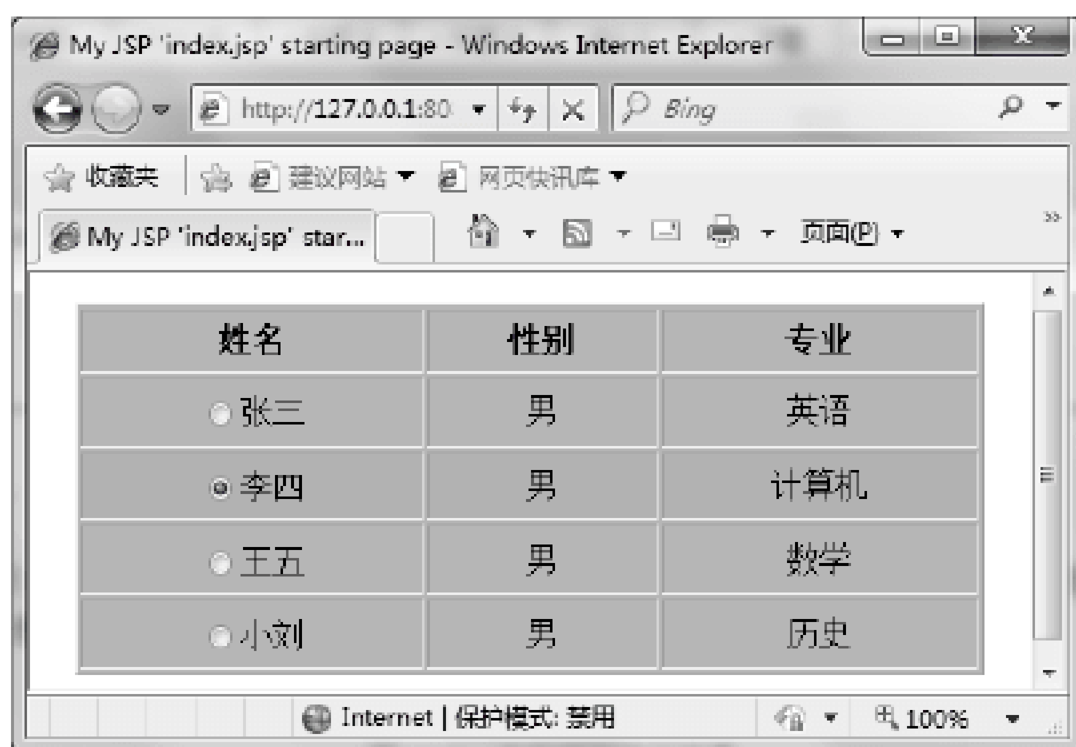


图 12.14　通过单选按钮控制表格的行高亮显示

关键技术

实现本实例的步骤如下：

（1）为表格行添加单击事件。

（2）为当前单选按钮所在的行添加高亮样式，并将单选按钮设置为选中状态。

本实例中使用了 end()方法，当前对象是$(this)。当进行 adClass("selected")操作时，对象并未发生变化；当执行 siblings().removeClass("selected")操作时，对象已经变为$(this).siblings()。因此，所有的操作都是针对$(this)获取的对象进行的。

设计过程

（1）在页面中定义表格显示信息，具体代码如下：

```
<table width="449" height="185" border="1" align="center" bgcolor="#66CCCC">
    <thead>
      <tr><th>姓名</th><th>性别</th><th>专业</th></tr>
    </thead>
    <tbody align="center">
        <tr><td> <input type="radio" name="radiobutton" value="radiobutton" />张三</td><td>男</td><td>英语</td></tr>
        <tr><td><input type="radio" name="radiobutton" value="radiobutton" />李四</td><td>男</td><td>计算机</td></tr>
        <tr><td><input type="radio" name="radiobutton" value="radiobutton" />王五</td><td>男</td><td>数学</td></tr>
        <tr><td><input type="radio" name="radiobutton" value="radiobutton" />小刘</td><td>男</td><td>历史</td></tr>

    </tbody>
</table>
```

（2）在页面中定义 CSS 样式，为表格添加选择的样式，具体代码如下：

```
<style type="text/css">
        .selected{
                background: #FF99CC;
        }
</style>
```

（3）在页面中定义 JavaScript 函数，实现为表格中的单元格添加样式，并设计表格中的单选按钮为选中状态，具体代码如下：

```
<script type="text/javascript">
        $(function() {
                $('tbody>tr').click(function(){                    //表格单击事件
                        $(this)
                        .addClass('selected')                      //为表格行添加样式
                        .siblings().removeClass('selected')        //移除其他行的样式
                        .end()
                        .find(':radio').attr('checked',true);      //设置单选按钮为选中状态
                });
        });
</script>
```

秘笈心法

心法领悟 322：简单方法实现为表格的行添加样式。

实现本实例还有一种简单的方法，代码如下：

```
$('table:radio:checked').parents("tr").addClass('selected');
```

上述代码的含义是通过向父节点逐步地添加样式。

实例 323 通过复选框控制表格的行高亮显示

光盘位置：光盘\MR\12\323

高级

实用指数：★★★★

实例说明

与单选按钮不同，复选框可以实现一次选择多行内容，并没有限制选择的行数。本实例实现通过复选框控制表格的行高亮显示，运行结果如图 12.15 所示。

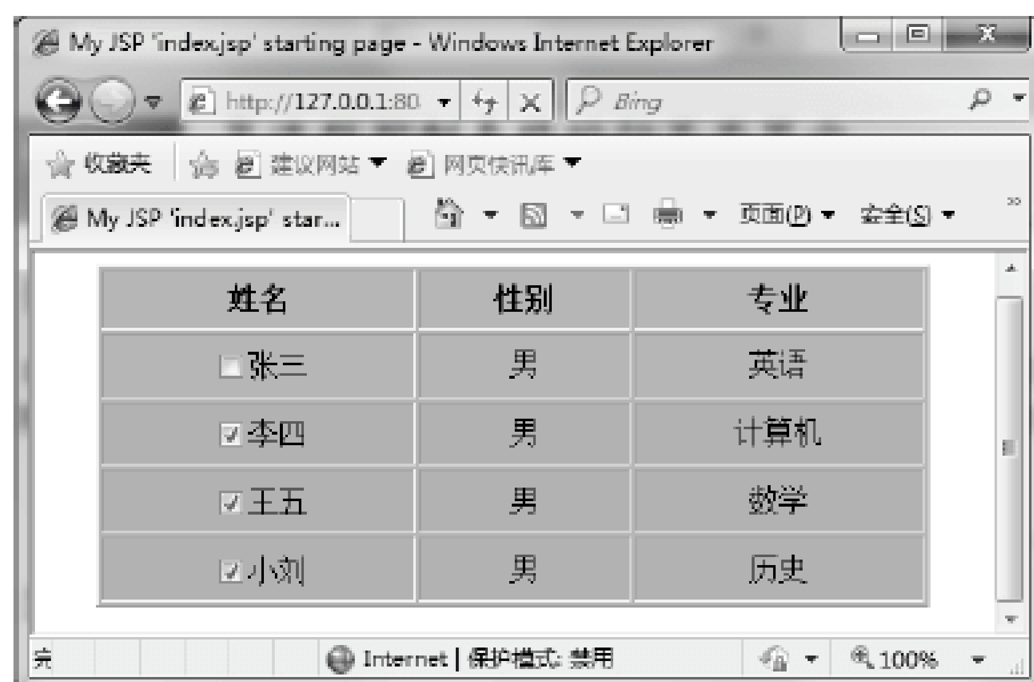


图 12.15 通过复选框控制表格的行高亮显示

关键技术

通过复选框控制表格的行的高亮样式，与通过单选按钮控制表格的行的高亮样式的技术类似。本实例中需要注意的是，如果单击的是已经添加高亮样式的表格行，则会去掉相应的样式；如果单击的是没有添加样式的内容，则将指定的样式添加到表格行中。判断表格行中是否添加了指定的高亮样式，可使用 hasClass()方法来实现。该方法的语法如下：

```
hasClass("cName")
```

参数说明

cName：进行判断的样式。

设计过程

在页面中定义 JavaScript 函数，实现判断用户单击的表格行是否有指定的高亮样式，如果有则将该样式删除，如果没有则添加样式。具体代码如下：

```
<script type="text/javascript">
        $(function() {
                $('tbody>tr').click(function(){                         //表格单击事件
                        if($(this).hasClass('selected')){               //如果表格中包含指定的样式
                                $(this).removeClass('selected')         //将该样式删除
                                .find(':checkbox').attr('checked',false);   //取消复选框的选中状态
                        }
                        else{                                           //如果表格没有指定样式
                                $(this).addClass('selected')            //向表格中添加样式
                                .find(':checkbox').attr('checked',true);    //设置复选框为选中状态
                        }
                });
        });
</script>
```

心法领悟 323：使用 jQuery 的三元运算符。

在 Java 程序中，可以使用三元运算符来代替 if…else 条件语句。在 jQuery 语句中同样可以使用三元运算符，例如，实例的代码使用三元运算符进行修改，代码可修改为：

```
$(this)[hasSelected?"removeClass":"addClass"]('selected')
```

实例 324　表格的展开与关闭

光盘位置：光盘\MR\12\324

高级

实用指数：★★★

实例说明

表格的展开与关闭主要针对的是父级行与子级行之间的关系，可以通过父级行来控制子级行的展开与关闭。本实例初始效果如图 12.16 所示，将文科类的学生行关闭后，页面的运行结果如图 12.17 所示。

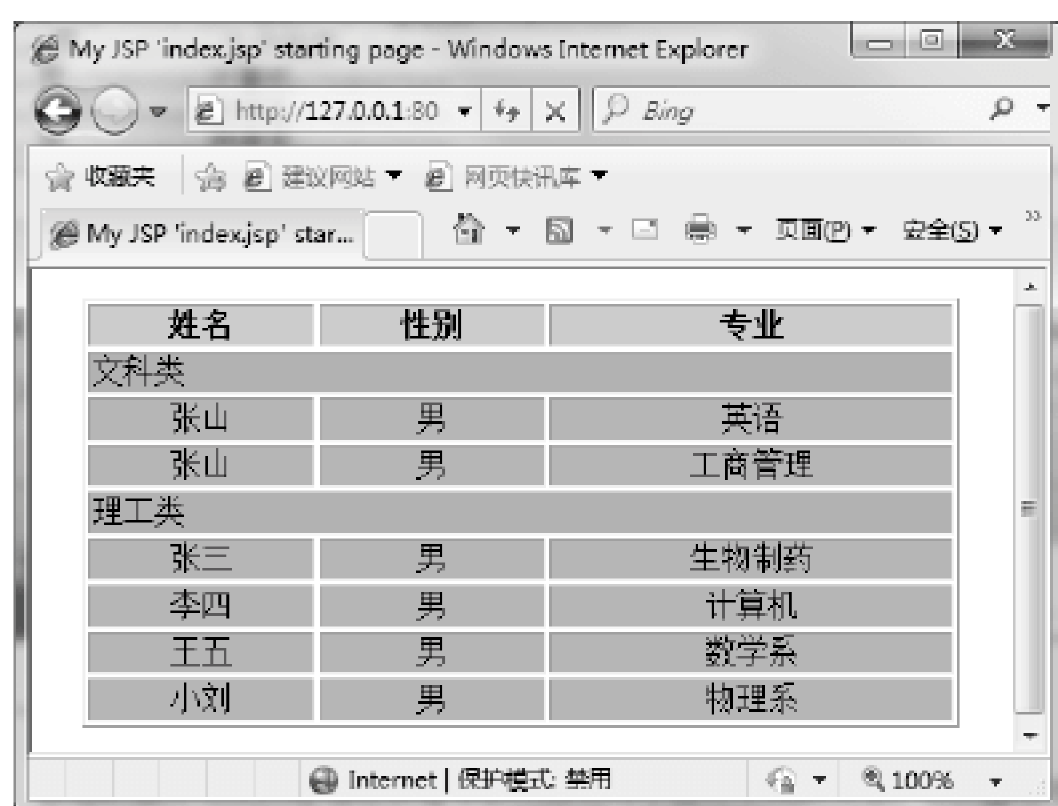


图 12.16　页面初始效果



图 12.17　关闭文科类学生行后的页面

关键技术

实现本实例，需要设置父级行与子级行的 id，通过控制表格的隐藏与显示来实现用户所看到的表格的展开与关闭。因此，在表格中给每个<tr>元素设置属性是非常重要的。本实例中每个父级行都设置了 class="parent" 属性，并设置了父级行的 id 值，而每个父级行对应的子级行，只设置了 class 属性，并且 class 的值是在 id 值的基础上通过加“child_”来设置。

（1）在页面中定义表格，分别将学生进行归类，分为文科类与理工类，并设置父级表格行与子级表格行的属性。具体代码如下：

```
<table width="449" height="185" border="1" align="center">
    <thead>
      <tr bgcolor="#CCCCCC"><th>姓名</th><th>性别</th><th>专业</th></tr>
    </thead>
    <tbody align="center">
        <tr align="left" class = "parent" id = "row01"><td colspan="3">文科类</td></tr>
        <tr class="child_row01"><td>张山</td><td>男</td><td>英语</td></tr>
        <tr class="child_row01"><td>张山</td><td>男</td><td>工商管理</td></tr>
        <tr align="left" class = "parent" id = "row02"><td colspan="3">理工类</td></tr>
        <tr class="child_row02" ><td>张三</td><td>男</td><td>生物制药</td></tr>
        <tr class="child_row02"><td>李四</td><td>男</td><td>计算机</td></tr>
        <tr class="child_row02"><td>王五</td><td>男</td><td>数学系</td></tr>
        <tr class="child_row02"><td>小刘</td><td>男</td><td>物理系</td></tr>
    </tbody>
</table>
```

（2）在页面中定义 JavaScript 函数，实现将用户单击的父级表格行对应的子级表格隐藏或显示。如果对应的子级表格行是显示状态则将其隐藏；如果对应的子级表格行是隐藏状态，则将其设置为显示状态。具体代码如下：

```
<script type="text/javascript">
        $(function() {
                $('tr.parent').click(function(){                    //判断是否单击了父级表格行
```

```
                    $(this)
                    .toggleClass("selected")                    //添加或删除表格样式
                    .siblings('.child_'+this.id).toggle();      //隐藏或显示父级表格行对应的子级表格行
               });
          });
</script>
```

秘笈心法

心法领悟 324：jQuery 中的选择器。

选择器是 jQuery 中最常用的技术，本章中的很多内容也是通过选择器实现的。例如，在网页中每个 id 名称只能使用一次，而 class 允许重复使用。如果使用基本选择器实现操作给定 id 匹配的元素，可以使用“$("#id")”；如果操作给定的类名匹配的元素，可以使用“$(".test")”；如果操作根据给定的元素名匹配元素，可以使用“$("p")”。

实例 325	利用文本框的值实现对表格内容的筛选 光盘位置：光盘\MR\12\325	中级 实用指数：★★★★

实例说明

本实例实现的是通过 jQuery 技术实现对表格内容的筛选。运行程序，在如图 12.18 所示页面中的“筛选姓”文本框中输入学生的姓，如“张”，单击“查询”按钮，即可将所有“张”姓学生信息显示出来，如图 12.19 所示。



图 12.18　学生信息页面

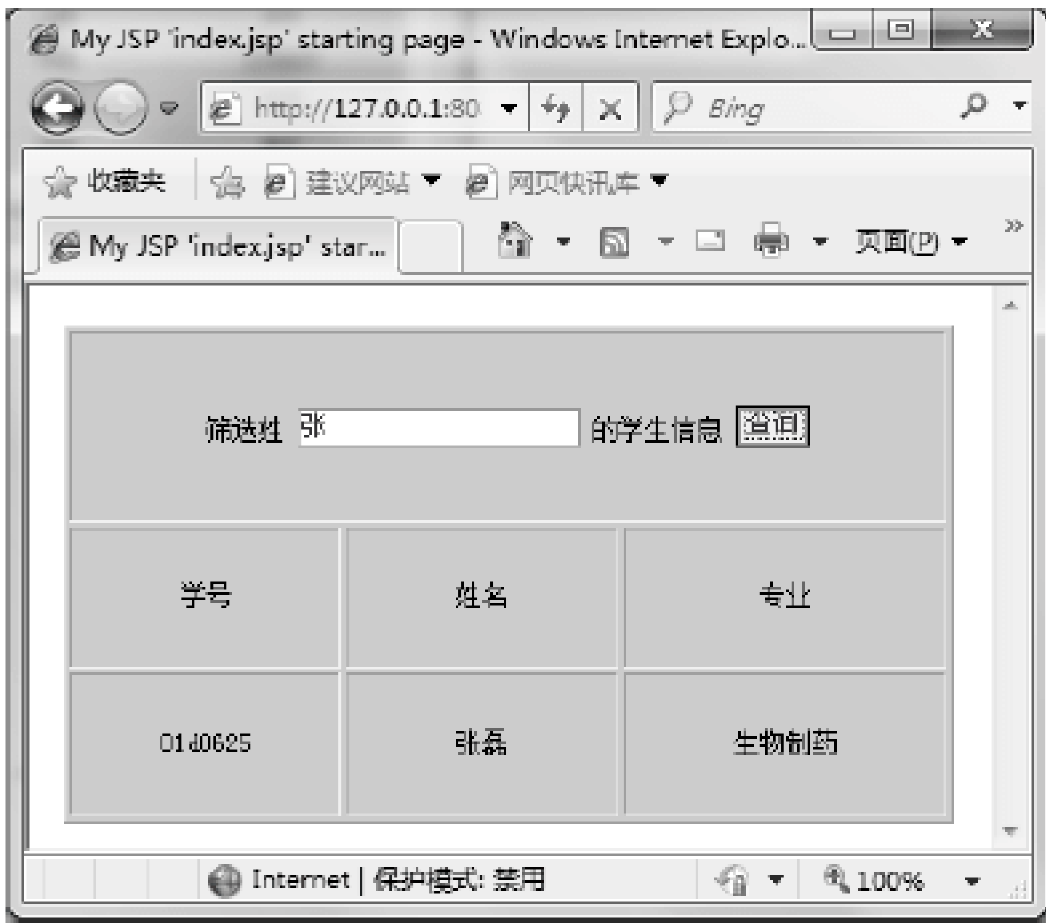


图 12.19　查询结果



本实例实现筛选表格中的内容，使用了 contains 选择器与 filter()方法。filter()属于一种筛选函数，用于筛选出与指定表达式匹配的元素集合。除此之外，jQuery 中还包含其他一些筛选函数，如表 12.2 所示。

表 12.2　jQuery 中的筛选函数及其说明

函　　数	说　　明
eq(index)	获取指定参数索引值位置上的元素，索引值从 0 开始计算
filter(fn)	筛选出与指定函数返回值匹配的元素集合
is(expr)	以一个表达式来检查当前选择的元素集合，如果有满足条件的元素则返回 true
map(callback)	将一组元素转换成其他数组
not(expr)	删除与指定表达式匹配的元素
add(expr)	把与表达式匹配的元素添加到 jQuery 对象中
contents()	查找匹配元素内部所有的子节点

■ 设计过程

（1）在页面中定义表格使用<thead>，使用<tbody>标签将表格头区分出来。具体代码参见配书光盘中的源程序。

（2）在页面中定义 JavaScript 函数，实现当用户单击页面中的“查询”按钮时，完成筛选工作。具体代码如下：

```
<script type="text/javascript">
    $(function(){
        $("#button").click(function(){
            $("table tbody tr").hide()                                    //将表格行内容隐藏
            .filter(":contains('"+($('#filterName').val())+"')").show();  //筛选与文本框中匹配的表格行添加到页面中
        });
    });

</script>
```

■ 秘笈心法

心法领悟 325：表单专用选择器。

jQuery 中的选择器是 jQuery 中非常重要的部分。由于表单对象比较特殊，很多表单域都共用同一个元素 input，这为快速选择特定表单域带来了困难。jQuery 定义了一组表单专用选择器，例如“:input”匹配所有 input、textarea、select 和 button 表单元素，“:text”匹配所有的单行文本框，“:password”匹配所有密码框，“:radio”匹配所有单选按钮，“:checkbox”匹配所有复选框，“:submit”匹配所有提交按钮。

12.4 其他特效

实例 326	制作网页选项卡 光盘位置：光盘\MR\12\326	中级 实用指数：★★★★

■ 实例说明

实现选项卡功能，不管是对于应用程序还是 Web 程序来说都是非常重要的。使用 Web 实现网页选项卡，原理比较简单，通过切换选项卡来显示不同的内容。本实例将演示如何制作网页选项卡，运行结果如图 12.20 所示。

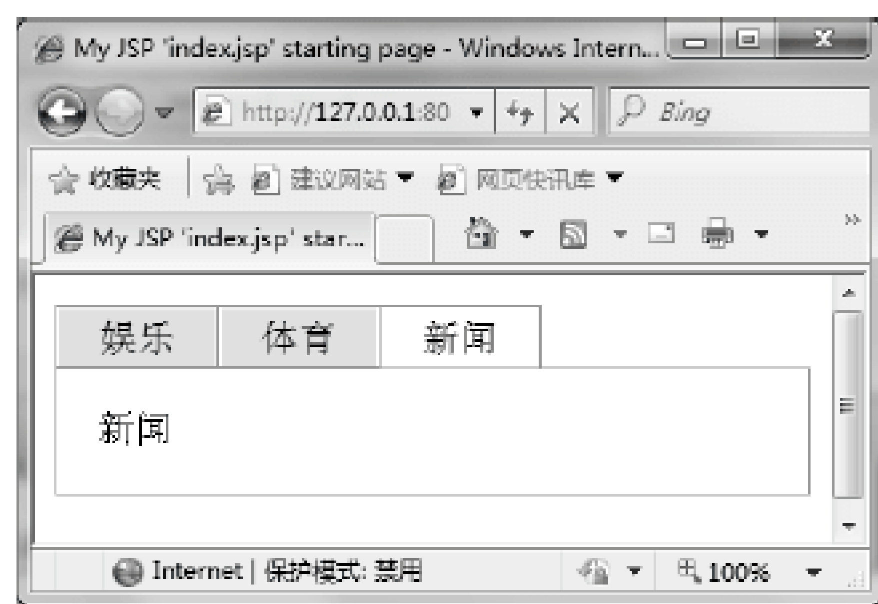


图 12.20　制作网页选项卡

■ 关键技术

要实现本实例功能，可以使用 addClass()方法与 removeClass()方法，当用户单击选项卡内容时，可通过<div>层的显示与隐藏来实现显示不同的内容。

■

（1）在页面中定义<div>层，实现使用<ul>与<li>标签定义页面显示内容。具体代码如下：

```
<ul class="tabs">
    <li><a href="#tab1">娱乐</a></li>
    <li><a href="#tab2">体育</a></li>
    <li><a href="#tab3">新闻</a></li>
</ul>
<div class="tab_container">
    <div id="tab1" class="tab_content">
        娱乐
    </div>
```

```
        <div id="tab2" class="tab_content">
            体育
        </div>
          <div id="tab3" class="tab_content">
            新闻
        </div>
</div>
```

（2）在页面中定义 CSS 样式，用于控制页面显示效果。具体代码参见配书光盘中的源程序。

（3）在页面中定义 JavaScript 函数，实现在页面中单击某选项卡显示特定的内容。具体代码如下：

```
<script type="text/javascript">
$(document).ready(function() {
        $(".tab_content").hide();                       //将表格内容隐藏
        $("ul.tabs li:first").addClass("active").show();  //将表格内容添加样式
        $(".tab_content:first").show();
        $("ul.tabs li").click(function() {               //判断用户是否单击了某选项卡
        $("ul.tabs li").removeClass("active");           //移除样式
        $(this).addClass("active");
        $(".tab_content").hide();
        var activeTab = $(this).find("a").attr("href");
        $(activeTab).fadeIn();
        return false;
        });
        });
</script>
```

秘笈心法

心法领悟 326：jQuery 的 4 个外部插入方法。

所谓外部插入，就是把内容插入到指定 jQuery 对象相邻元素内。与内部插入操作基本类似，外部插入包含 4 个方法。其中，after(content)用于在每个匹配的元素之后插入内容；before(content)用于在每个匹配的元素之前插入内容；insertAfter(content)用于把所有匹配的元素插入到另一个指定的元素或元素集合的后面；insertBefore(content)用于把所有匹配的元素插入到另一个指定的元素或元素集合的前面。

实例 327	日期拾取器 光盘位置：光盘\MR\12\327	高级 实用指数：★★★★

实例说明

用户注册会员时，表单中大多数都需要输入出生日期、毕业日期或者工作日期等。对于日期的输入，目前基本上存在两种形式，一种是直接手动输入，另一种是弹出一个日期拾取器，在其中选择相应的日期。本实例通过使用 jQuery 插件创建日期拾取器，实现选择日期的功能，如图 12.21 所示。



图 12.21　日期拾取器

本实例是通过 jQuery 插件 datepick 实现的，读者可以打开 jQuery 的官方网站 http://www.jquery.com 下载该插件和 jQuery 框架。解压缩后，将 jquery.datepick.js、jquery.datepick-zh-CN.js 和 jquery.1.3.2.min.js 复制到项目根目录的 JS 文件夹下。同时，将压缩包中的 redmond.datepick. css 样式文件也一并复制到 JS 文件夹下（压缩包中还有很多样式文件，读者可以自己选择使用哪个样式）。然后在 Web 页中引入这些文件，之后初始化日期选择器插件 datepick，代码如下：

```
$('#txtdate').datepick({ dateFormat: 'yyyy-mm-dd'});
```

设计过程

（1）在 JSP 中添加一个 HTML 的文本框，用于显示选择后的日期。具体代码参见配书光盘中的源程序。

（2）在页面中引入 jQuery 框架、jQuery 插件 datepick 和相应的样式，代码如下：

```
<script type="text/javascript" src="js/jquery.1.3.2.min.js"></script>
<script type="text/javascript" src="js/jquery.datepick.js"></script>
<script type="text/javascript" src="js/jquery.datepick-zh-CN.js"></script>
<link href="js/redmond.datepick.css" rel="stylesheet" type="text/css" />
```

（3）初始化 datepick 插件，并且通过 dateFormat: 'yyyy-mm-dd'设置文本框中显示的日期格式。代码如下：

```
<script type="text/javascript">
$(function() {
    $('#txtdate').datepick({ dateFormat: 'yyyy-mm-dd'});          //初始化 datepick 插件，设置显示的日期格式
});
</script>
```

（4）在后台代码中创建一个 getvalue()方法，用于返回选择的日期。代码如下：

```
public string getvalue()                                        //该方法用于在后台获取前台 HTML 控件的值
{
    System.Collections.Specialized.NameValueCollection nv = new System.Collections.Specialized.NameValueCollection(System.Web.HttpContext.
Current.Request.Form);
    return nv.GetValues("tdate")[0].ToString();                  //获取文本框中显示的值
}
```

（5）在“注册”按钮的 Click 事件中获取注册信息，其中包括使用 getvalue()方法获取选择的日期。代码如下：

```
if (getvalue() != "")                                           //如果 Web 用户控件选择了日期
{
string info ="用户名："+txtUserName.Text + "/密码：" + txtPwd.Text + "/年龄：" + txtAge.Text + "/性别：" + rdbSex.SelectedValue + "/生日：" +
getvalue();                                                     //获取输入的基本信息及选择的日期
    ClientScript.RegisterStartupScript(this.GetType(), "", "alert('" + info+ "');", true);   //弹出信息，实际开发中可以将这些信息插入到数据库中
}
else                                                            //如果返回值为空
{
    ClientScript.RegisterStartupScript(this.GetType(), "", "alert('请选择生日');", true);   //说明没有选择日期，弹出提示信息
}
```

秘笈心法

心法领悟 327：获取系统盘中的 Windows 路径。

在实际开发过程中，有时可能需要获取系统盘的 Windows 路径。那么该如何获取这个路径呢？通过 JavaScript 脚本可以轻松实现，关键代码如下：

```
var fso = new ActiveXObject("Scripting.FileSystemObject");
var tfolder = fso.GetSpecialFolder(0);
alert(tfolder)
```

实例 328 网页软键盘

光盘位置：光盘\MR\12\328

高级

实用指数：★★★

实例说明

随着网络的普及，网上交易已经逐渐成为一种潮流。例如，在网上购物、网上转账等。大多数银行的网上银行系统在登录时都提供了一个虚拟的网页键盘，以避免通过键盘输入，防止黑客攻击，从而增强了安全性。本实例通过 jQuery 插件实现网页软键盘功能，运行结果如图 12.22 所示。



图 12.22　网页软键盘

本实例是通过 jQuery 插件 keypad 实现的，读者可以打开 jQuery 的官方网站 http://www.jquery.com 下载该插件和 jQuery 框架。解压缩后，将 jquery.keypad.js 和 jquery.1.3.2.min.js 复

制到项目根目录的 JS 文件夹下。同时，将压缩包中的 jquery.keypad.alt.css 样式文件也一并复制到 JS 文件夹下（压缩包中还有其他样式文件，读者可以自己选择使用哪个样式）。然后在 Web 页中引入这些文件，之后初始化虚拟键盘插件 keypad。代码如下：

```
        $(function () {
        $('#defaultKeypad').keypad({prompt:'',keypadOnly:false,layout:$.keypad.qwertyLayout});
});
```

设计过程

（1）在项目的 index.jsp 页面中引入 jQuery 框架、jQuery 插件 keypad 和相应的样式，代码如下：

```
<link href="js/jquery.keypad.alt.css" rel="stylesheet" type="text/css" />
<script type="text/javascript" src="js/jquery.1.3.2.min.js"></script>
<script type="text/javascript" src="js/jquery.keypad.js"></script>
```

（2）初始化 keypad 插件，并且设置 keypadOnly 和 layout 属性，实现在虚拟键盘中显示字母、其他字符和数字，以便用户能够选择。代码如下：

```
<script type="text/javascript">
$(function () {
        //初始化 keypad 插件
        $('#defaultKeypad').keypad({prompt:'',keypadOnly:false,layout:$.keypad.qwertyLayout});
});
</script>
```

（3）在“登录”按钮的 Click 事件中获取输入的用户名和密码，其中输入的密码是通过 getvalue()方法获取的。代码如下：

```
if (getvalue() != "")                                        //如果返回值不为空，说明通过软键盘输入了数据
{
    //获取输入的信息，包括登录名和通过软键盘输入的密码
    string info ="登录名："+txtUserName.Text+"登录密码："+getvalue();
    //弹出对话框显示输入的信息，实际开发中可以将这些数据插入数据库
    ClientScript.RegisterStartupScript(this.GetType(), "", "alert('"+info+"');", true);
}
else                                                         //否则
{
    //弹出提示信息
    ClientScript.RegisterStartupScript(this.GetType(), "", "alert('请输入密码');", true);
}
```

秘笈心法

心法领悟 328：正确获取 Servlet 的设置信息。

要获取 Servlet 程序在 Web.xml 文件中的设置信息，必须使用该 Servlet 在 Web.xml 文件中所映射的 URL 路径来访问它，而不能使用激活器的方式进行访问。

实例 329	图片幻灯片 光盘位置：光盘\MR\12\329	高级 实用指数：★★★

实例说明

日常浏览网页时，图片幻灯片的实例随处可见。例如，在 QQ 空间相册中，就有图片幻灯片的效果。此外，在某些网站的首页上也会有图片幻灯片。这个功能到底是如何实现的呢？其实使用 jQuery 插件实现该功能是非常方便的，本实例就是用 jQuery 插件开发一个图片幻灯片，运行结果如图 12.23 所示。



图 12.23　本实例的运行结果

本实例是通过 jQuery 插件 jcarousel 实现的，读者可以打开网站 http://billwscott.com/carousel/或者直接在百度、谷歌搜索下载该插件和 jQuery 框

架。解压缩后，将 jquery-1.2.3.pack.js、jquery.jcarousel.pack.js、jquery.jcarousel.css 和 tango/skin.css 复制到项目根目录的 JS 文件夹下，然后在 Web 页中引入这些文件，之后初始化 jcarousel 插件。代码如下：

```
<script type="text/javascript">
jQuery(document).ready(function() {
    jQuery('#mycarousel').jcarousel({
        start: 3
    });
});
</script>
```

设计过程

（1）在页面中引入 jQuery 框架、jQuery 插件 jcarousel 和相应的样式，代码如下：

```
<link href="js/style.css" rel="stylesheet" type="text/css" />
<script type="text/javascript" src="js/jquery-1.2.3.pack.js"></script>
<script type="text/javascript" src="js/jquery.jcarousel.pack.js"></script>
<link rel="stylesheet" type="text/css" href="js/jquery.jcarousel.css" />
<link rel="stylesheet" type="text/css" href="js/tango/skin.css" />
```

（2）初始化 jQuery 插件 jcarousel，代码如下：

```
<script type="text/javascript">
jQuery(document).ready(function() {
    jQuery('#mycarousel').jcarousel({
        start: 3
    });
});
</script>
```

（3）在<body></body>区域中加入
标签，并设置其 id 属性，然后在该标签之间加入幻灯片要显示的图片。这样当运行网站时，就可以实现图片幻灯片的效果。代码如下：

```
<ul id="mycarousel" class="jcarousel-skin-tango">
<li><img src="image/01.jpeg" width="75" height="75" alt="" /></li>
<li><img src="image/02.jpeg" width="75" height="75" alt="" /></li>
<li><img src="image/03.jpeg" width="75" height="75" alt="" /></li>
<li><img src="image/04.jpeg" width="75" height="75" alt="" /></li>
<li><img src="image/05.jpeg" width="75" height="75" alt="" /></li>
<li><img src="image/06.jpeg" width="75" height="75" alt="" /></li>
</ul>
```

秘笈心法

心法领悟 329：jQuery 提供的替换结构。

jQuery 提供了 replaceWith(content)和 replaceAll(selector)方法来实现 HTML 结构替换。其中，replaceWith()能够将所有匹配的元素替换成指定的 HTML 或 DOM 元素。

实例 330 颜色拾取器

光盘位置：光盘\MR\12\330

中级

实用指数：★★★★

实例说明

提到颜色拾取器，可能有些读者不太明白是什么意思。下面举例说明。例如，使用 QQ 聊天时，如果想设置字体颜色，可以在 QQ 聊天窗口中打开设置字体颜色的对话框。在这个对话框中设置颜色的界面就是颜色拾取器。本实例将通过 jQuery 实现一个简单的颜色拾取器，运行结果如图 12.24 所示。

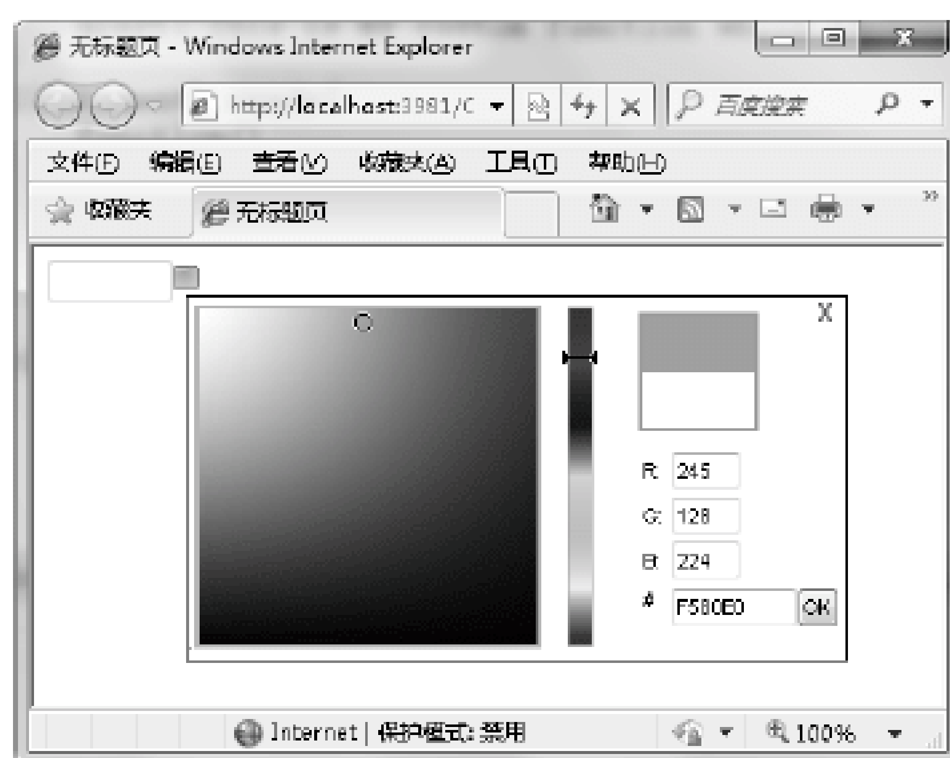


图 12.24　颜色拾取器

关键技术

本实例是通过 jQuery 插件 ColorPicker 实现的，读者可以打开网站 http://interface.eyecon.ro 或者直接在百度、谷歌搜索下载该插件

和 jQuery。解压缩后，将 jQuery 文件夹复制到项目根目录的 JS 文件夹下，然后在 Web 页中引入这些文件，之后初始化插件 ColorPicker。代码如下：

```
<script language="javascript">
        function myokfunc(){
                alert("This is my custom function which is launched after setting the color");
        }
        $(document).ready(
                function()
                {
                        $.ColorPicker.init();
                }
        );
</script>
```

设计过程

（1）在页面中引入 jQuery 框架、jQuery 插件 ColorPicker 和相应的样式，代码如下：

```
<script src="js/jquery/jquery.js" type="text/javascript"></script>
        <script src="js/jquery/ifx.js" type="text/javascript"></script>
        <script src="js/jquery/idrop.js" type="text/javascript"></script>
        <script src="js/jquery/idrag.js" type="text/javascript"></script>
        <script src="js/jquery/iutil.js" type="text/javascript"></script>
        <script src="js/jquery/islider.js" type="text/javascript"></script>
        <script src="js/jquery/color_picker/color_picker.js" type="text/javascript"></script>
        <link href="js/jquery/color_picker/color_picker.css" rel="stylesheet" type="text/css"/>
```

（2）在<body></body>区域中加入一个 div，并命名为 myshowcolor，其作用是当单击此 div 时，弹出颜色拾取器，该 div 的 HTML 代码如下：

```
<div id="myshowcolor" style="border-width: 1px; border-color: black; width:15px; height:15px; background-image: url('color.png');"> </div>
```

（3）在后台代码中创建一个 getvalue()方法，用户选择颜色的 RGB 值。代码如下：

```
<script type="text/javascript">
        function myokfunc(){
                alert("This is my custom function which is launched after setting the color");
        }
        $(document).ready(
                function()
                {
                        $.ColorPicker.init();
                }
        );
</script>
```

秘笈心法

心法领悟 330：本地影像视频 AVI 格式。

AVI 的英文全称为 Audio Video Interleaved，即音频视频交错格式。所谓“音频视频交错”，就是可以将视频和音频交织在一起进行同步播放。这种视频格式的优点是图像质量好，可以跨多个平台使用；其缺点是体积过于庞大，而且更加糟糕的是压缩标准不统一，最普遍的现象就是高版本 Windows 媒体播放器播放不了采用早期编码编辑的 AVI 格式视频，而低版本 Windows 媒体播放器又播放不了采用最新编码编辑的 AVI 格式视频。

实例 331	广告轮显 光盘位置：光盘\MR\12\331	中级 实用指数：★★★★

在很多商务网站中，首页中经常会看到广告图片轮流显示。这不仅为网站增加了动态效果，同时也使网站获得了可观的利润。那么这个广告图片轮流显示是如何开发的呢？当然，网上有很多这方面的 JavaScript 脚本，不过本实例笔者是通过 jQuery 框架插件 easyslide 实现的，其特点是功能强大，制作简单。本实例运行结果如

图 12.25 所示。



图 12.25　广告轮显

关键技术

本实例是通过 jQuery 插件 easyslide 实现的，读者可以打开网站 http://www.ezjquery.com/cgi-bin/webapp.rb?r=access&lan=gb 或者直接在百度、谷歌搜索下载该插件。解压缩后，将 jquery-1.2.3.pack.js 和 jquery.myslide.js 复制到项目根目录的 JS 文件夹下，然后在 Web 页中引入这些文件，之后初始化广告轮显插件 easyslide。代码如下：

```
<script>
$(document).ready(function(){
$.init_slide('imgstore','showhere',1,0,0,1,5000,1);

});
</script>
```

设计过程

（1）在页面中引入 jQuery 框架、jQuery 插件 easyslide，代码如下：

```
<script type="text/javascript" src="js/jquery-1.2.3.pack.js"></script>
<script type="text/javascript" src="js/jquery.myslide.js"></script>
```

（2）初始化 jQuery 插件 easyslide，代码如下：

```
<script type="text/javascript">
$(document).ready(function(){
$.init_slide('imgstore','showhere',1,0,0,1,5000,1);
});
</script>
```

（3）在<body></body>区域中加入一个 div，id 属性设置为 showher。该 div 将作为图片轮显区域，其 HTML 代码如下：

```
<div id="showher" align="left" ></div>
```

（4）紧接着，再添加一个 div，命名为 imgstore，然后在该 div 中加入想要轮显的图片。这样当运行网站后，该 div 区域中的图片就会在页面中轮流显示。代码如下：

```
<div id="imgstore" style="display:none">
<img src="01.jpeg" title="盘古开天之处"/>
<img src="02.jpeg" title="笔架山神路" />
<img src="03.jpeg" title="海滩美景 1"/>
<img src="04.jpeg" title="海滩美景 2"/>
<img src="05.jpeg" title="海滩美景 3"/>
<img src="06.jpeg" title="海滩美景 4"/>
</div>
```

心法领悟 331：本地影像视频 nAVI 格式。

nAVI 是 newAVI 的缩写，是一个名为 ShadowRealm 的“地下组织”发展起来的一种新视频格式。它是由

Microsoft ASF 压缩算法修改而来的，但是又与网络影像视频中的 ASF 视频格式有所区别，它以牺牲原有 ASF 视频文件视频“流”特性为代价，而通过增加帧率来大幅提高 ASF 视频文件的清晰度。

实例 332	图片放大镜 光盘位置：光盘\MR\12\332	中级 实用指数：★★★★

实例说明

在一些购物网站中搜索商品时，搜索结果中通常提供的是缩略图，有时无法看清。不过，将鼠标指针放到商品图片上时，就会显示放大的图片，使商品浏览起来更加清晰。其实这也算作一种简单的图片放大镜效果。本实例要实现的功能就是通过 jQuery 插件开发图片放大镜，运行结果如图 12.26 所示。

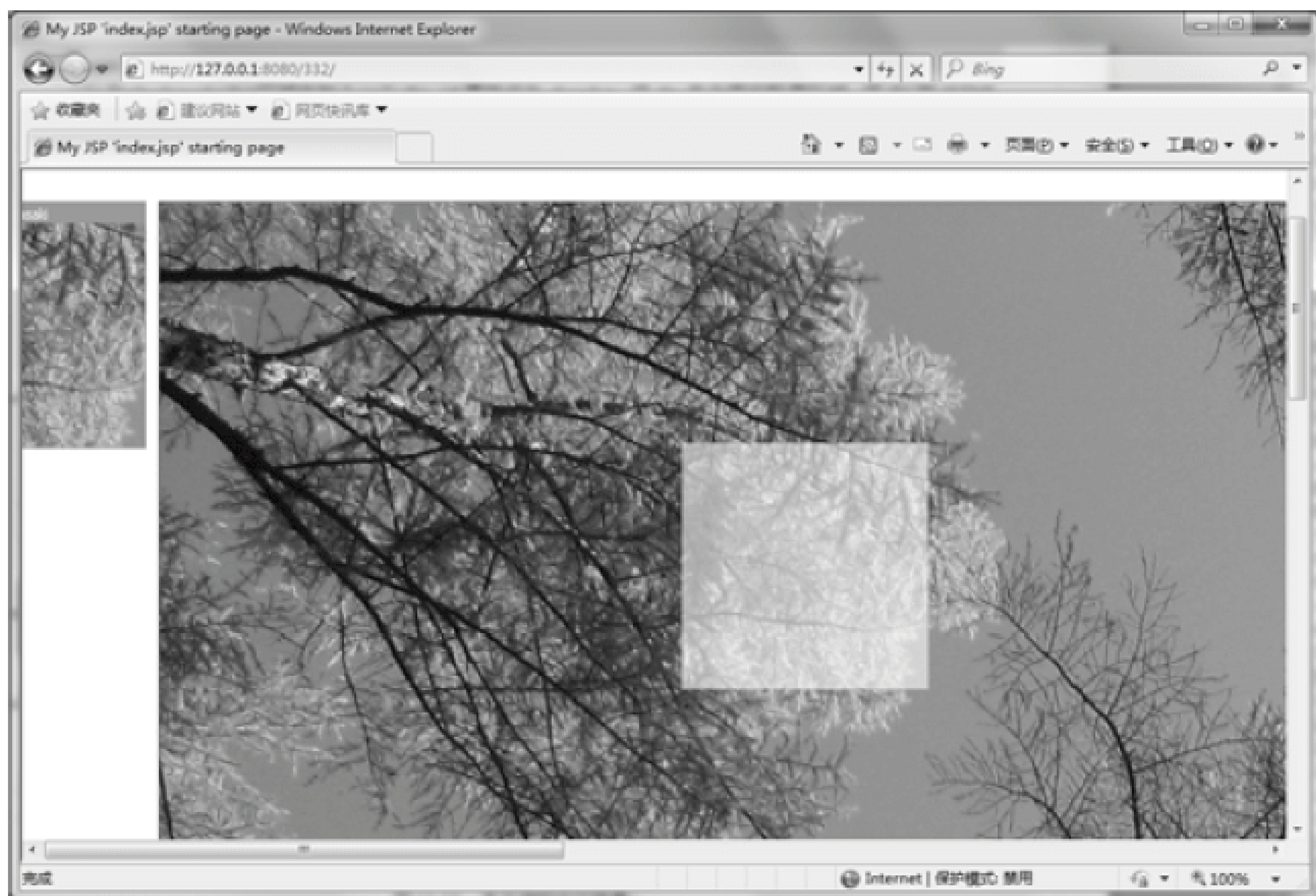


图 12.26 图片放大镜

关键技术

本实例是通过 jQuery 插件 jqzoom 实现的，读者可以打开网站 http://www.mind-projects.it/projects/jqzoom/archives/jqzoom_ev1.0.1.zip 或者直接在百度、谷歌搜索下载该插件。解压缩后，将 JS 和 CSS 文件夹复制到项目根目录下，然后在 Web 页中引入相应的.js 和.css 文件，之后初始化插件 jqzoom。代码如下：

```
<script type="text/javascript">
$(function() {
        $(".jqzoom").jqzoom();
});
</script>
```

（1）在页面中引入 jQuery 框架、jQuery 插件 jqzoom 和相应的样式，代码如下：

```
<script src="js/jquery-1.3.2.min.js" type="text/javascript"></script>
<script src="js/jqzoom.pack.1.0.1.js" type="text/javascript"></script>
<link rel="stylesheet" href="css/jqzoom.css" type="text/css"/>
```

（2）初始化 jQuery 插件 jqzoom，以便运行网站后能够实现图片放大镜效果。代码如下：

```
<script type="text/javascript">
$(function() {
        $(".jqzoom").jqzoom();
});
</script>
```

（3）在<body></body>区域中加入一个 div，id 属性设置为 content。在该 div 中，首先通过<a href= "CIMG0927.JPG" class="jqzoom" style="" title="kawasaki">设置原图，然后通过<img src="2.jpeg" title=

"kawasakigreen" style="border: 1px solid #666;"/>加载缩略图。这样，当鼠标指针停留在缩略图某个位置时，该位置的原图便会显示出来，从而实现局部放大的效果。代码如下：

```
<div id="content" style="margin-top:100px;margin-left:100px; height: 230px; width: 592px; margin-right: 0px;">
<a href="CIMG0927.JPG" class="jqzoom" style="" title="kawasaki">
<img src="2.jpeg" title="kawasakigreen" style="border: 1px solid #666;"/>
</a>
</div>
```

秘笈心法

心法领悟 332：jQuery 中的 css()方法。

css()方法可实现为页面元素定义样式，或者获取指定属性的值。例如，css(name)方法能够获取匹配元素中第一个元素的指定属性的属性值。

实例 333 文本编辑器

光盘位置：光盘\MR\12\333

中级

实用指数：★★★★

实例说明

对于文本编辑器，相信读者不会感到陌生（如 Word 也算作文本编辑器）。文本编辑器的应用非常广泛，例如，发表留言、发表论坛帖子、写邮件等，但是，如何在网页中添加文本编辑器呢？本实例将通过 jQuery 插件开发漂亮的文本编辑器，运行结果如图 12.27 所示。

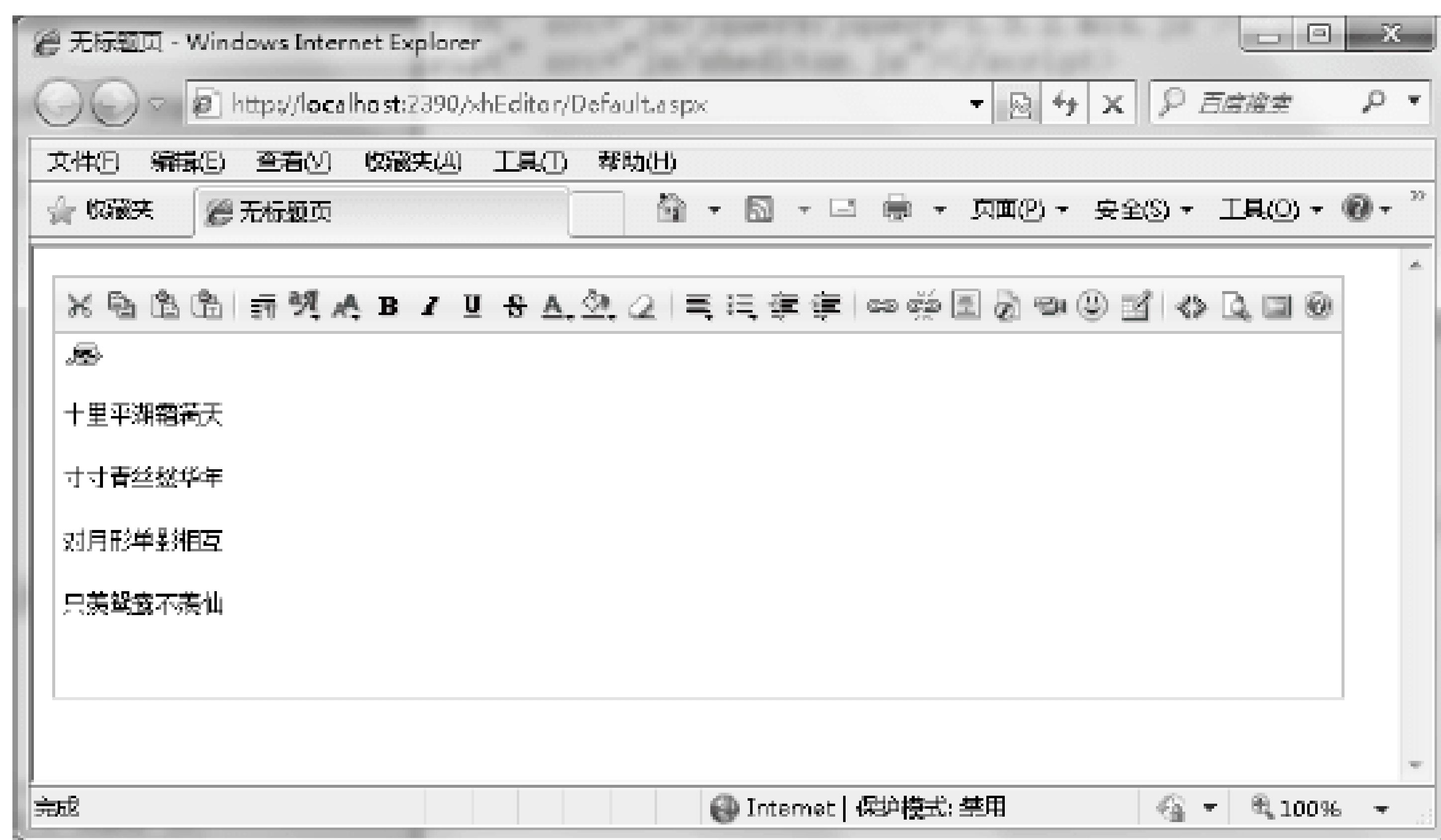


图 12.27　文本编辑器

关键技术

本实例是通过 jQuery 插件 xhEditor 实现的，读者可以打开网站 http://code.google.com/p/xheditor/downloads/list 或者直接在百度、谷歌搜索下载该插件。下载之后，将 xheditor.js 、 xheditor_emot 、 xheditor_plugins 和 xheditor_skin 复制到项目根目录的 JS 文件夹下，然后在 Web 页中引入相应的.js 和.css 文件。代码如下：

```
<link rel="stylesheet" href="js/common.css" type="text/css" media="screen" />
<script type="text/javascript" src="js/jquery/jquery-1.3.2.min.js"></script>
<script type="text/javascript" src="js/xheditor.js"></script>
```

设计过程

（1）在页面中引入 jQuery 框架、jQuery 插件 xhEditor 和相应的样式。代码如下：

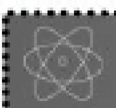
```
<link rel="stylesheet" href="js/common.css" type="text/css" media="screen" />
<script type="text/javascript" src="js/jquery/jquery-1.3.2.min.js"></script>
<script type="text/javascript" src="js/xheditor.js"></script>
```

（2）在<body></body>区域中加入一个 HTML 控件 textarea，并将其 class 属性设置为 xheditor，以便其在运行时能够加载 jQuery 插件 xhEditor 的样式。这样，在运行网站时，就能看到 xhEditor 插件的漂亮外观了。代码如下：

```
<textarea id="elm1" name="elm1" class="xheditor" rows="12" cols="80" style="width: 80%">
```

心法领悟 333：jQuery 中的位移方法。

jQuery 定义了一个非常实用的方法——offset()来实现位移操作。该方法能够获取匹配元素的第一个元素在当前窗口的坐标，坐标以窗口左上顶点为圆点进行参考。返回对象包含 top 和 left 属性值，分别表示该元素距离左侧和顶部的距离。



实例 334

右键菜单

光盘位置：光盘\MR\12\334

高级

实用指数：★★★★

实例说明

读者对右键菜单应该不会感到陌生，无论是操作系统中，还是一些应用程序中，都会有右键菜单。但是，在网站中的右键菜单都是 IE 浏览器默认的右键菜单。如果想自定义右键菜单，应该如何实现呢？本实例主要通过 jQuery 插件来实现一个简单的右键菜单，运行结果如图 12.28 所示。

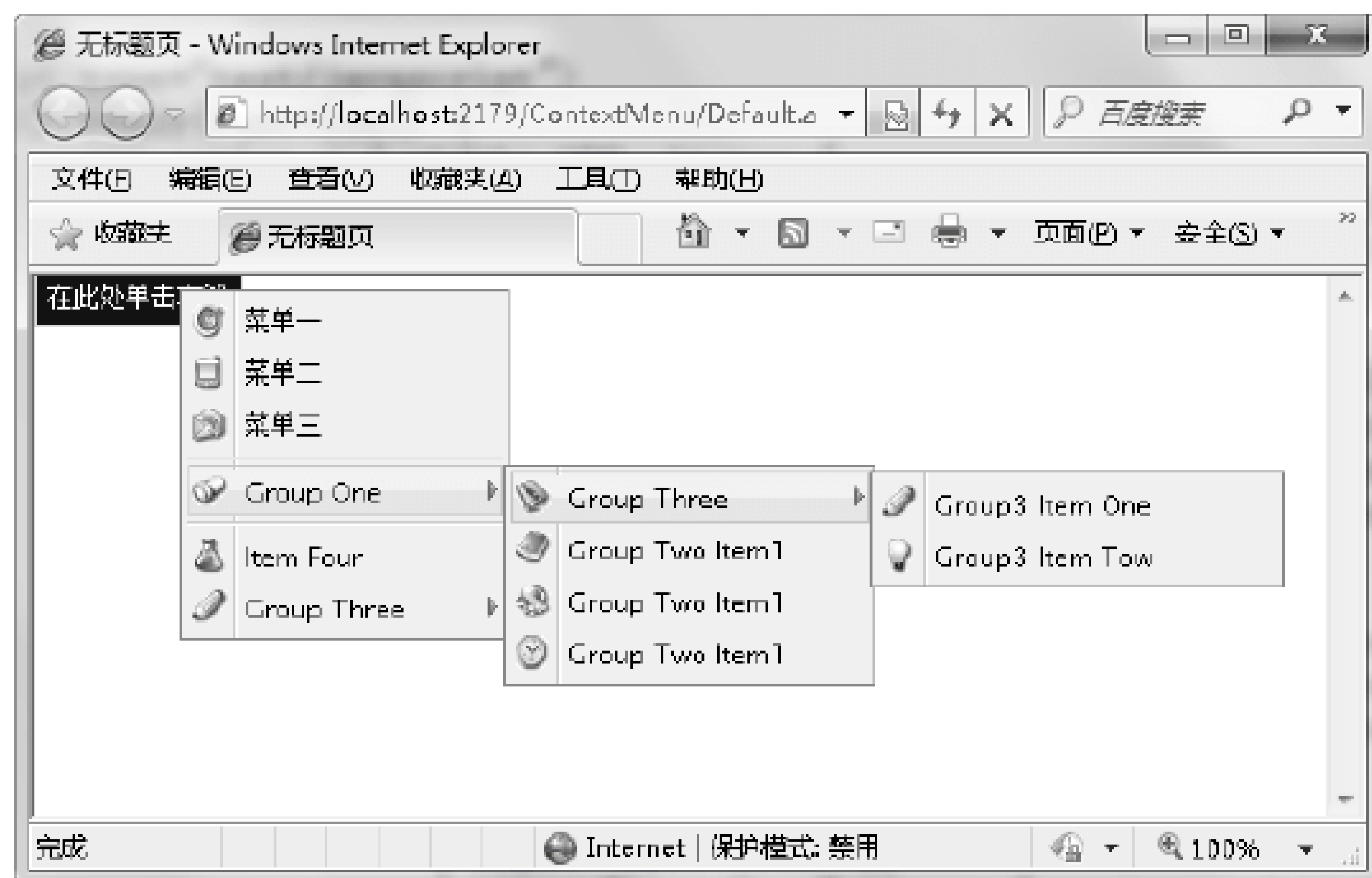


图 12.28　右键菜单

本实例是通过 jQuery 插件 ContextMenu 实现的，读者可以打开网站 http://www.web-delicious.com/jquery-plugins-demo/wdContextMenu.zip 或者直接在百度、谷歌搜索下载 ContextMenu 插件。解压缩后，将 css、sample-css 和 src 文件夹复制到项目根目录下。然后在 Web 页中引入这些文件，初始化插件 ContextMenu。代码如下：

```
<script type="text/javascript">
        $().ready(function() {
            var option = { width: 150, items: [
                                { text: "菜单一", icon: "sample-css/wi0126-16.gif", alias: "1-1", action: menuAction },
                                { text: "菜单二", icon: "sample-css/ac0036-16.gif", alias: "1-2", action: menuAction },
                                { text: "菜单三", icon: "sample-css/ei0021-16.gif", alias: "1-3", action: menuAction },
                                { type: "splitLine" },
                                { text: "Group One", icon: "sample-css/wi0009-16.gif", alias: "1-4", type: "group", width: 170, items: [
                                    { text: "Group Three", icon: "sample-css/wi0054-16.gif", alias: "2-2", type: "group", width: 190, items: [
                                        { text: "Group3 Item One", icon: "sample-css/wi0062-16.gif", alias: "3-1", action: menuAction },
                                        { text: "Group3 Item Tow", icon: "sample-css/wi0063-16.gif", alias: "3-2", action: menuAction }
                                    ]
                                    },
                                    { text: "Group Two Item1", icon: "sample-css/wi0096-16.gif", alias: "2-1", action: menuAction },
                                    { text: "Group Two Item1", icon: "sample-css/wi0111-16.gif", alias: "2-3", action: menuAction },
                                    { text: "Group Two Item1", icon: "sample-css/wi0122-16.gif", alias: "2-4", action: menuAction }
                                ]
                                },
                                { type: "splitLine" },
                                { text: "Item Four", icon: "sample-css/wi0124-16.gif", alias: "1-5", action: menuAction },
                                { text: "Group Three", icon: "sample-css/wi0062-16.gif", alias: "1-6", type: "group", width: 180, items: [
                                    { text: "Item One", icon: "sample-css/wi0096-16.gif", alias: "4-1", action: menuAction },
                                    { text: "Item Two", icon: "sample-css/wi0122-16.gif", alias: "4-2", action: menuAction }
                                ]
                                }
                                ], onShow: applyrule,
                onContextMenu: BeforeContextMenu
            };
            function menuAction() {
                alert(this.data.alias);
            }
            function applyrule(menu) {
                if (this.id == "target2") {
                    menu.applyrule({ name: "target2",
                        disable: true,
                        items: ["1-2", "2-3", "2-4", "1-6"]
                    });
                }
                else {
```

```
                    menu.applyrule({ name: "all",
                        disable: true,
                        items: []
                    });
                }
            }
            function BeforeContextMenu() {
                return this.id != "target3";
            }
            $("#target ").contextmenu(option);
        });
</script>
```

（1）在页面中引入 jQuery 框架、jQuery 插件 ContextMenu 和相应的样式。代码如下：

```
<link href="sample-css/page.css" rel="stylesheet" type="text/css" />
<link href="css/contextmenu.css" rel="stylesheet" type="text/css" />
<style type="text/css">
.target
{
        border:solid 1px #ffccee;
        padding:5px;
        background-color:Blue;
        color:#fff;
        display:inline;
}
</style>
<script src="src/jquery.js" type="text/javascript"></script>
<script src="src/Plugins/jquery.contextmenu.js" type="text/javascript"></script>
```

（2）初始化 jQuery 插件 ContextMenu，在初始化过程中设置右键菜单的数量和名称，以及设置是否包含二级子菜单等。代码如下：

```
<script type="text/javascript">
        $().ready(function() {
            var option = { width: 150, items: [
                            { text: "菜单一", icon: "sample-css/wi0126-16.gif", alias: "1-1", action: menuAction },
                            { text: "菜单二", icon: "sample-css/ac0036-16.gif", alias: "1-2", action: menuAction },
                            { text: "菜单三", icon: "sample-css/ei0021-16.gif", alias: "1-3", action: menuAction },
                            { type: "splitLine" },
                            { text: "Group One", icon: "sample-css/wi0009-16.gif", alias: "1-4", type: "group", width: 170, items: [
                                { text: "Group Three", icon: "sample-css/wi0054-16.gif", alias: "2-2", type: "group", width: 190, items: [
                                    { text: "Group3 Item One", icon: "sample-css/wi0062-16.gif", alias: "3-1", action: menuAction },
                                    { text: "Group3 Item Tow", icon: "sample-css/wi0063-16.gif", alias: "3-2", action: menuAction }
                                ]
                                },
                                { text: "Group Two Item1", icon: "sample-css/wi0096-16.gif", alias: "2-1", action: menuAction },
                                { text: "Group Two Item1", icon: "sample-css/wi0111-16.gif", alias: "2-3", action: menuAction },
                                { text: "Group Two Item1", icon: "sample-css/wi0122-16.gif", alias: "2-4", action: menuAction }
                            ]
                            },
                            { type: "splitLine" },
                            { text: "Item Four", icon: "sample-css/wi0124-16.gif", alias: "1-5", action: menuAction },
                            { text: "Group Three", icon: "sample-css/wi0062-16.gif", alias: "1-6", type: "group", width: 180, items: [
                                { text: "Item One", icon: "sample-css/wi0096-16.gif", alias: "4-1", action: menuAction },
                                { text: "Item Two", icon: "sample-css/wi0122-16.gif", alias: "4-2", action: menuAction }
                            ]
                            }
                            ], onShow: applyrule,
                onContextMenu: BeforeContextMenu
            };
            function menuAction() {
                alert(this.data.alias);
            }
            function applyrule(menu) {
                if (this.id == "target2") {
                    menu.applyrule({ name: "target2",
                        disable: true,
                        items: ["1-2", "2-3", "2-4", "1-6"]
```

```
                });
            }
            else {
                menu.applyrule({ name: "all",
                    disable: true,
                    items: []
                });
            }
        }
        function BeforeContextMenu() {
            return this.id != "target3";
        }
        $("#target").contextmenu(option);
    });
</script>
```

（3）在<body></body>区域中加入一个 div，id 属性设置为 target，实现在该 div 区域右击时，弹出设置的右键菜单。代码如下：

```
<div id="target"   class="target">在此处单击右键</div>
```

秘笈心法

心法领悟 334：jQuery 中的动画效果。

JavaScript 并没有提供设计动画效果的函数，不过 jQuery 弥补了这一不足。它一方面把平时常用的简单动画封装为直接调用的方法，另一方面还定义了几个比较实用的动画方法，调用这些方法即可快速实现各种复杂的动画效果。例如，show()方法用于显示隐藏的匹配元素；hide()方法用于隐藏显示的元素。

实例 335	结合 jQuery 实现在线裁剪 光盘位置：光盘\MR\12\335	高级 实用指数：★★★★

实例说明

现在大多数论坛都提供了设置会员头像的功能，当上传头像图片后，程序会提供这样一个功能，就是可以裁剪图片的某个区域作为个性头像。当然，现在也有很多网站允许网友上传图片，然后在线裁剪出某一块，裁剪之后还提供下载等功能，非常方便。下面就一起来学习一下应如何实现图片在线裁剪功能。本实例运行结果如图 12.29 所示。



图 12.29　图片在线裁剪

关键技术

（1）开发思路

图片裁剪，就是在图片中剪切出某一部分。裁剪图片之前，要通过鼠标绘制出选区。该过程由于是在客户端实现的，因为之前已经介绍了 jQuery 框架，所以应该考虑使用 JavaScript 来实现。笔者搜索了一下相关的插件，找到了 Jcrop 插件。该插件可以在某张图片上拖出一个选区，并获取选区左上角坐标、宽和高等信息，有了这些信息就可以裁剪图片了。

（2）难点解析

要使用 Jcrop 插件，就必须了解该插件的一些常用属性，如表 12.3 所示。

表 12.3　Jcrop 插件的常用属性及说明

属　　性	说　　明	属　　性	说　　明
bgColor	设置背景色	minSelect	设置最小选区
bgOpacity	设置背景透明度	maxSize	设置选择最大值
borderOpacity	设置边框透明度	minSize	设置选择最小值
boxWidth	设置图片显示的宽度	watchShift	是否监视 Shift
boxHeight	设置图片显示的高度		

这些常用的属性中，比较值得注意的是 boxWidth 和 boxHeight 属性。上传图片时，图片的分辨率可能很大，如果不加以限制，就会破坏整个页面的布局。此时 boxWidth 和 boxHeight 属性起作用，通过这两个属性，可以将大分辨率的图片等比例缩放到指定的范围内。例如，在本例中就用到了这两个属性，代码如下：

```
$(function() { $('#oImage').Jcrop({ boxWidth: 520, boxHeight: 330, onChange: showCoords, onSelect: showCoords }); });
```

了解完属性，再来看看该插件的两个重要事件——onChange 和 onSelect 事件。其中，onChange 事件是当选区更改时回调该事件；而 onSelect 事件是当设置选区之后回调该事件。

（1）进入 jQuery 官方网站 http://www.jquery.com，下载 Jcrop 插件。

（2）下载之后，解压缩，将 example 文件夹下的 css 和 js 文件夹复制到网站项目根目录下。Jcrop 插件解压缩后的目录如图 12.30 所示。

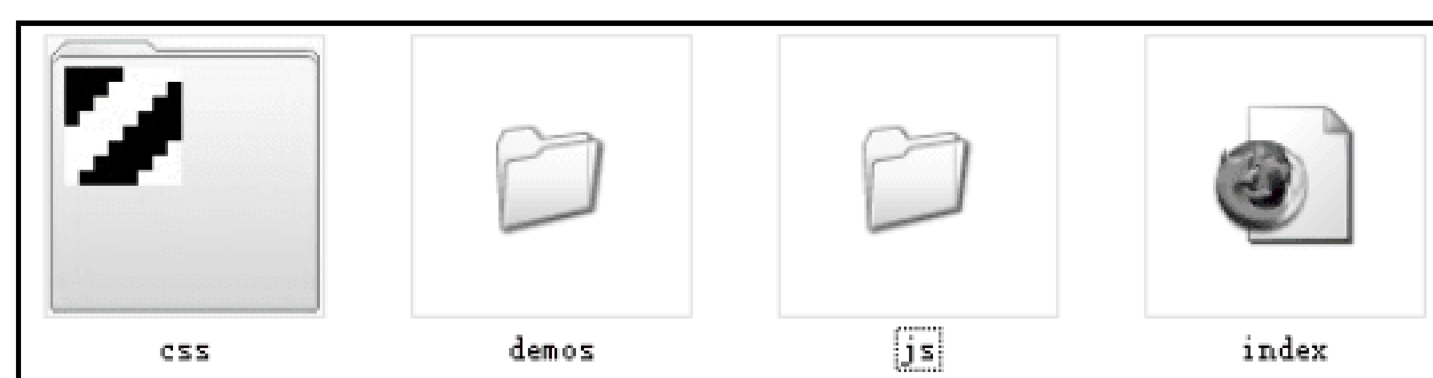


图 12.30　Jcrop 插件解压缩后的目录

（3）复制到根目录下之后，在 Default.aspx 页的 HTML 代码中首先要引入 Jcrop 插件所需的.js 文件及 CSS 样式。代码如下：

```
<script type="text/javascript" src="js/jquery.pack.js"></script>
<script type="text/javascript" src="js/jquery.Jcrop.pack.js"></script>
<link rel="stylesheet" href="css/jquery.Jcrop.css" type="text/css" />
```

（4）向页面中添加 1 个<img>图像、3 个 ImageButton 按钮、1 个 FileUpload 控件、4 个文本框。在 HTML 代码的<head></head>标签中，初始化 Jcrop 插件，使其对<img>图像进行操作。当通过鼠标拖出选区后，会执行回调，触发 onChange 事件。指定该事件调用 showCoords()函数，获取选区左上角的坐标和选区的宽度、高度。通过 checkCoords()函数检查是否设置选区，通过 sendImg()函数设置裁剪预览图的高度和宽度，并且将图片的地址、选区左上角坐标以及选区的高度和宽度等信息传递给 Handler.ashx 进行裁剪。代码如下：

```
<script type="text/javascript">
            $(function() { $('#oImage').Jcrop({ boxWidth: 520, boxHeight: 330, onChange: showCoords, onSelect: showCoords }); });
                                                            //当选择、改变选区时都执行 showCoords()函数
      function showCoords(c) {
            $("#txtX").val(c.x);                            //得到选中区域左上角横坐标
            $("#txtY").val(c.y);                            //得到选中区域左上角纵坐标
            $("#txtW").val(c.w);                            //得到选中区域的宽度
            $("#txtH").val(c.h);                            //得到选中区域的高度
      }
      function checkCoords() {
            var defaulturl=document.getElementById("oImage").src;
            defaulturl=defaulturl.substring(defaulturl.lastIndexOf("/")+1);
            if (defaulturl == "default1.jpg") {
                  alert("请上传图片");
                  return false;
            }
            else {
                  if (parseInt($('#txtH').val()) && parseInt($('#txtW').val())) {
                        sendImg();
                        return true;
                  }
                  else {
                        alert("请设置裁剪区域");
```

```
                    return false;
                }
            }
        };
        function sendImg() {
         var p = document.getElementById("oImage").src;
         var x = document.getElementById("txtX").value;
         var y = document.getElementById("txtY").value;
         var w = document.getElementById("txtW").value;
         var h = document.getElementById("txtH").value;
         var ow = 222;
         var oh = 300;
         var rate = w / h;
         if (rate > 1)                                   //选区的宽大于高
         {
             document.getElementById("imgCreat").width = ow;
             document.getElementById("imgCreat").height = ow / rate;
         }
         else if (rate < 1)                              //选区的高大于宽
         {
             document.getElementById("imgCreat").width = oh * rate;
             document.getElementById("imgCreat").height = oh;
         }
         else if (rate == 1) {
             document.getElementById("imgCreat").width = 222;
             document.getElementById("imgCreat").height =222;
         }
         document.getElementById("imgCreat").src = "Handler.ashx?p=" + p + "&x=" + x + "&y=" + y + "&w=" + w + "&h=" + h + "&" +
Math.random();
     }
</script>
```

注意：此处要对文件名进行 UTF8 编码；否则，如果文件名是中文，则当用户下载时会出现乱码的问题。

（5）在项目中新建一个“一般处理程序”，命名为 Handler.ashx，用于接收首页传递过来的图片路径、左上角坐标、选区的宽度和高度，然后在该文件中将原图按照这些参数进行裁剪，将裁剪后的图片保存到服务器上，以便用户下载。代码如下：

```
public void ProcessRequest (HttpContext context) {
        int x = Convert.ToInt32(context.Request["x"]);                 //获取裁剪区的 x 坐标
        int y = Convert.ToInt32(context.Request["y"]);                 //获取裁剪区的 y 坐标
        int dropWidth = Convert.ToInt32(context.Request["w"]);         //裁剪区域的宽度
        int dropHeight = Convert.ToInt32(context.Request["h"]);        //裁剪区域的高度
        string oPath = Convert.ToString(context.Request["p"]);         //原图片路径
        oPath = HttpContext.Current.Server.MapPath("UpLoad")+"//"+System.IO.Path.GetFileName(oPath);
        context.Response.ContentType = "image/jpeg";
        cutImage(oPath, x, y, dropWidth, dropHeight).WriteTo(context.Response.OutputStream);
}
```

（6）在上述代码中，调用 cutImage()方法进行裁剪，该方法将裁剪后的图片保存到 user 文件夹中，通过将裁剪后的图片存储到 MemoryStream 中，即可获取剪裁后的图片。代码如下：

```
public System.IO.MemoryStream cutImage(string oPath, int x, int y, int width, int height)
{
     System.Drawing.Bitmap bm = new System.Drawing.Bitmap( oPath);           //创建画板
     System.Drawing.Rectangle rg = new System.Drawing.Rectangle(x, y, width, height);
     System.Drawing.Imaging.PixelFormat format = bm.PixelFormat;
     System.Drawing.Bitmap nbm = bm.Clone(rg, format);
     string sPath = HttpContext.Current.Server.MapPath("user");               //获取文件路径
     System.IO.MemoryStream ms2 = new System.IO.MemoryStream();               //创建内存流
     nbm.Save(ms2, System.Drawing.Imaging.ImageFormat.Jpeg);                  //将文件保存到内存流
     if (!System.IO.Directory.Exists(sPath))                                  //如果文件夹不存在
     {
         System.IO.Directory.CreateDirectory(sPath);                          //创建文件夹
     }
     string newImageName = "HCDY" + DateTime.Now.Year + DateTime.Now.Month + DateTime.Now.Day + DateTime.Now.Hour + DateTime.
Now.Minute + DateTime.Now.Second + DateTime.Now.Millisecond +System.IO.Path.GetExtension(oPath); //设置文件在服务器端的文件名
     //保存文件到服务器端
```

```
        nbm.Save(sPath+"\\"+newImageName,System.Drawing.Imaging.ImageFormat.Jpeg);
        //将文件的相对路径写入 Cookie 中
        HttpContext.Current.Response.Cookies["url"].Value = "user//"+newImageName;
        bm.Dispose();
        nbm.Dispose();
        return ms2;
}
```

秘笈心法

心法领悟 335：开发手记。

开发本例时，首要考虑的就是如何能够在图片上拖曳出选区、如何能够获取选区相对原图的左上角坐标和选区宽度、高度。幸好 jQuery 提供了一个完美的插件——Jcrop。这个插件提供了很多属性和回调事件，用户可以很好地设置各项功能，而且拖曳出的选区很漂亮，同时还能获取裁剪图片所需的各项数据。既然 Jcrop 能够做到这些，那么开发起来就简单多了。因此，笔者选择通过该插件结合 JSP，开发出图片在线裁剪程序。

12.5 对 Ajax 的支持

实例 336	检测用户名是否被占用 光盘位置：光盘\MR\12\336	高级 实用指数：★★★

实例说明

在实现用户注册功能时，为了避免用户名的重复，应该在用户将注册信息提交之前，对用户输入的用户名进行检查，只有当系统中不存在当前注册的用户名时，才允许提交注册信息。运行本实例，在用户注册页面中输入用户名，然后单击“检测”超链接，即可检测当前的用户名是否重复，如果重复则将弹出提示对话框，如图 12.31 所示。

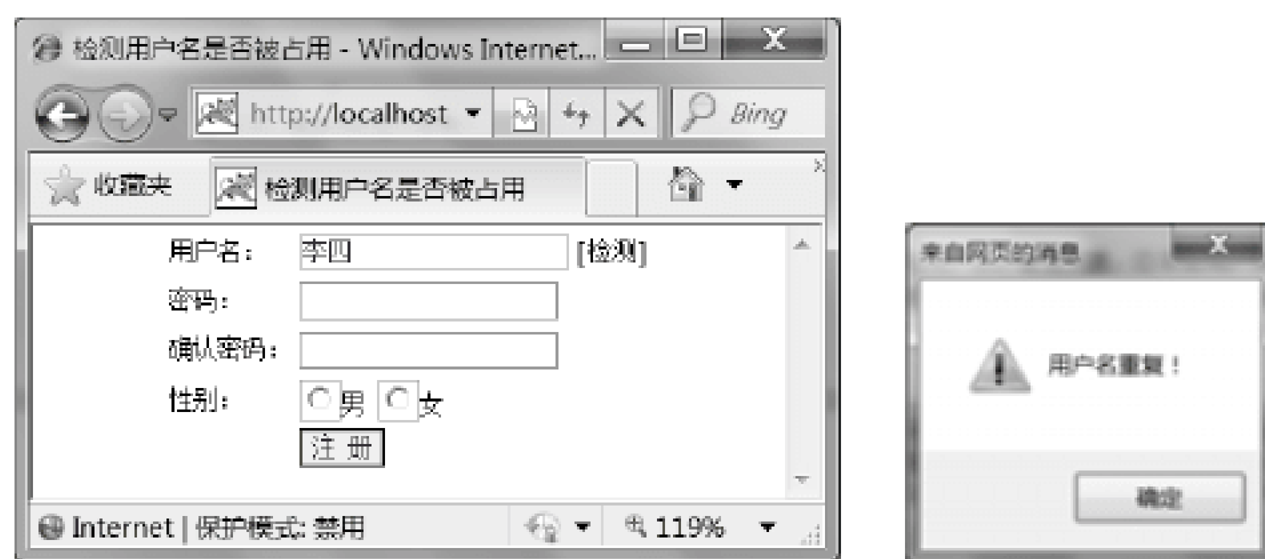


图 12.31　检测用户名是否被占用

关键技术

jQuery 框架不仅是一个非常好用的 JS 类库，还实现了对 Ajax 的封装。本实例就是应用 jQuery 中提供的$.ajax()来异步提交用户名到服务器中，然后根据这个用户名来查询数据库中是否存在重复，最后将结果以 JSON 的格式返回给客户端，$.ajax()方法再根据回调函数获取到返回的 JSON 结果，根据该结果来判断用户名是否重复。$.ajax()方法的语法如下：

```
$.ajax( {
        url : 'ValidateServlet',                    //请求的 URL 路径
        type : 'POST',                              //提交方式
        data : $('#loginForm').serialize(),         //提交的数据，整个表单
        dataType : 'json',                          //返回 JSON 数据
        success : function(data) {
                ……
        }
});
```

参数说明

❶ url：String 类型，表示发送请求的地址。

❷ type：String 类型，表示请求方式（GET 或 POST）。默认请求方式为 GET。

❸ data：类型为 Object 或 String，表示要发送到服务器的数据。

❹ dataType：指定服务器返回的数据类型。可用的类型有 xml、html、json、text 等。

❺ success：类型为 Function，表示请求成功后调用的回调函数。该函数有两个参数，date 为服务器返回的数据，并根据 dataType 设置的类型进行处理；textStatus 参数表示描述状态的字符串。

■

（1）创建 UserDao 类，在该类中编写根据用户名查询用户信息的方法 findUserByUserName()，如果查询出重名用户则返回 true。代码如下：

```
public boolean findUserByUserName(String userName){
        Statement stmt = null;
        Connection con = null;
        boolean res = false;
        try{
                con = getConn();
                String sql = "select id from tb_user where username ='"+userName+"'";
                stmt = con.createStatement();
                ResultSet rs = stmt.executeQuery(sql);
                if(rs.next()){
                        res = true;
                }
        }catch(Exception ex){
                ex.printStackTrace();
        }
        ……
        return res;
}
```

（2）创建 validate.js 文件，在该文件中编写 validateUser()方法，通过 jQuery 的$.ajax()方法异步提交用户名的数据。代码如下：

```
function validateUser(){
 var userName = $.trim($('#name').val());//获取到用户名文本框的值
 if(!userName){
        alert('用户名不能为空！');
        return null;
 }
 $.ajax( {
        url : 'ValidateServlet',
        type : 'POST',
        data : $('#registerForm').serialize(),
        dataType : 'json',
        success : function(data) {
                if (data.success == false) {
                        alert(data.msg);
                        return;
                }else{
                        alert(data.msg);
                        return;
                }
        }
 });
}
```

（3）创建 index.jsp 页面，该页面中包含一个用户注册的表单，并且在“用户名”文本框之后包含一个<a>超链接，当单击该超链接时，将执行 validate.js 的 validateUser()方法提交用户名信息。关键代码如下：

```
<input type="text" name="name" id="name" />
<a href="#" onclick="validateUser()">[检测]</a>
```

（4）创建 ValidateServlet 类，在 doGet()方法中接收通过$.ajax()提交过来的用户名信息，然后查询是否存在重复，并返回 JSON 结果。代码如下：

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        String userName = request.getParameter("name");
        boolean res = UserDao.getInstance().findUserByUserName(userName);
        if(res){
                JSONKit.outJSONInfo("{success:false,msg:'用户名重复！'}", response);
        }else{
```

```
                JSONKit.outJSONInfo("{success:true,msg:'此用户名可以注册！'}", response);
        }
}
```

秘笈心法

心法领悟 336：jQuery 提交请求的其他方法。

$.ajax()是 jQuery 发送请求的最底层的方法，除了这个方法外，通过 jQuery 的其他方法同样可以发送请求数据。例如，$.get()方法以 GET 请求方式发送数据，$.post()方法以 POST 请求方式发送数据。这些方法的详细说明参见 jQuery 提供的相关 API。

实例 337 验证用户登录

光盘位置：光盘\MR\12\337

中级

实用指数：★★★

实例说明

在 Web 网站或企业级 Web 应用系统中，用户管理功能是必不可少的，其中便包含验证用户登录。为了带给用户更好的体验，本实例通过 jQuery 的 Ajax 框架来异步验证用户的登录。运行本实例，在如图 12.32 所示页面中输入用户名和密码后单击“登录”按钮，如果用户名或密码错误，将弹出提示信息，否则会跳转到登录成功页面。



图 12.32　验证用户登录

关键技术

在通过 jQuery 的$.ajax()方法实现异步提交时，data 参数值可以是 Object 对象或普通的 String 字符串。在提交表单时，可以将整个表单对象作为 data 进行提交。例如，本实例中设置表单的 id 为 loginForm 后，即可通过$("loginForm")获取表单对象作为 data 的参数提交了，并通过 serialize()方法对表单进行序列化。

（1）创建 UserDao 类，编写根据用户名和密码验证用户登录的方法 validateLogin()。代码如下：

```
public boolean validateLogin(String userName,String pwd){
        Statement stmt = null;
        Connection con = null;
        boolean res = false;
        try{
                con = getConn();
                String sql = "select id from tb_user where username ='"+userName+"' and password='"+pwd+"'";
                stmt = con.createStatement();
                ResultSet rs = stmt.executeQuery(sql);
                if(rs.next()){
                        res = true;
                }
        }catch(Exception ex){
                ex.printStackTrace();
        }finally{
                ……
        }
        return res;
}
```

（2）创建 index.jsp 页面，在该页面中添加用户名和密码文本框的登录表单。代码如下：

```
<form id="loginForm" method="post" action="ValidateServlet">
      <table>
        <tr>
                <td>用户名：</td>
                <td><input type="text" name="name" id="name" /></td>
```

```
        </tr>
        <tr>
                <td>密码：</td>
                <td><input type="password" name="pwd" id="pwd" /></td>
        </tr>
        <tr>
                <td> </td>
                <td><input type="button" value="登录" onclick="validateLogin()" /> <a href="#">注册</a> </td>
        </tr>
    </table>
</form>
```

（3）创建 validate.js 文件，编写 validateLogin()方法，在该方法中通过 jQuery 的$.ajax()方法异步请求服务器，提交表单数据，并通过回调方法验证用户登录。代码如下：

```
function validateLogin(){
 var userName = $.trim($('#name').val());        //获取到用户名文本框的值
 var pwd = $('#pwd').val();
 if(!userName){
        alert('用户名不能为空！');
        return null;
 }
 if(!pwd){
        alert('密码不能为空！');
        return null;
 }
 $.ajax( {
        url : 'ValidateServlet',                 //请求的 URL 路径
        type : 'POST',                           //提交方式
        data : $('#loginForm').serialize(),      //提交的数据，整个表单
        dataType : 'json',                       //返回 JSON 数据
        success : function(data) {
                if (data.success == false) {
                        alert(data.msg);
                        return;
                }
                location.href = 'success.jsp';
        }
 });
}
```

（4）创建 ValidateServlet 类，在 doPost()方法中获取请求参数的用户名和密码，然后通过查询数据库的方法验证用户登录，并返回 JSON 结果。代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        String userName = request.getParameter("name");                          //用户名
        String pwd = request.getParameter("pwd");                                //密码
        boolean res = UserDao.getInstance().validateLogin(userName,pwd);         //验证登录
        if(!res){
                JSONKit.outJSONInfo("{success:false,msg:'用户名或密码错误！'}", response);  //返回 JSON 数据
        }else{
                request.getSession().setAttribute("user", userName);             //保存到 Session
                JSONKit.outJSONInfo("{success:true,msg:'可以登录！'}", response);   //返回 JSON 数据
        }
}
```

秘笈心法

心法领悟 337：JSON 文件。

之所以会出现 JSON 这种数据格式的文件，主要是因为 XML 文档体积大且难于解析。JSON 文件和 XML 文档一样，可以方便地重用。而且，JSON 文件非常简洁，也容易阅读。JSON 的语法格式非常严格，构建的 JSON 文件必须完整无误，任何一个括号的不匹配或缺少逗号，都会导致页面的脚本终止运行。一个正确的 JSON 格式的文件如下：

```
{key1:value1,key2:value2,key3:value3}
```



实例 338 基于 jQuery 的 Ajax 聊天室

光盘位置：光盘\MR\12\338

中级

实用指数：★★★★

实例说明

本实例将通过 jQuery 的 Ajax 框架实现一个简单的聊天室。运行程序，在如图 12.33 所示页面中输入昵称和消息内容，然后单击“发送”按钮，即可实时看到聊天信息。

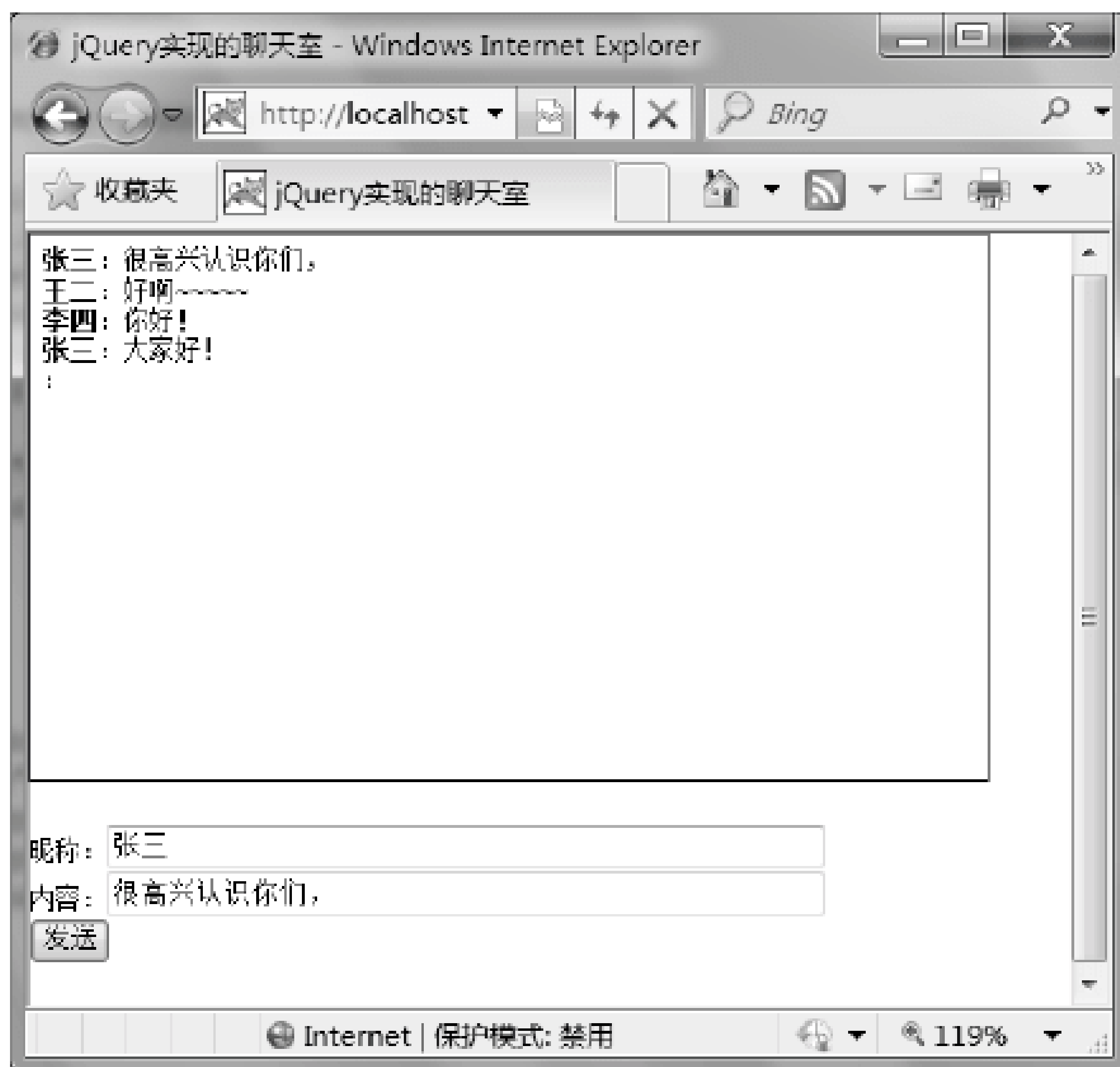


图 12.33　基于 jQuery 的 Ajax 聊天室

关键技术

本实例主要是使用 jQuery 的$.post()方法来实现聊天室，首先通过$.post()方法将文本框的聊天信息提交给服务器，然后在服务器端构建 XML 文档结果进行返回，最后通过$.post()的回调方法解析 XML 文档数据，再展现在页面中。$.post()方法的语法如下：

```
$.post(url [,data] [,callback] [,type])
```

参数说明

❶ url：String 类型，请求的 URL 地址。

❷ data：Object 类型，就是要发送到服务器的数据，为可选参数。

❸ callback：Function 类型，载入成功时回调函数自动将请求结果和状态传递给该方法，为可选参数。

❹ type：String 类型，服务器端返回的内容的格式，包括 xml、html、script、json、text 和_default，为可选参数。

（1）创建 index.jsp 页，添加发送信息的表单和显示聊天信息的窗口。代码如下：

```
<body>
<div id="wrapper">
        <p id="messagewindow"><span id="loading">加载中……</span></p>
        <form id="chatform">
                昵称：<input type="text" id="user" size="50" /><br />
                内容：<input type="text" id="msg" size = "50" ><br />
                <input type="submit" value="发送" /><br />
        </form>
</div>
</body>
```

（2）通过 jQuery 的$.post()编写请求服务器的方法，发送聊天信息。代码如下：

```
function updateMsg(){
```

```
        $.post("MessageServlet",{
                        message:$('#msg').val(),
                        name:$('#user').val(),
                        time:timestamp
        },function(xml){
                $('#loading').remove();
                addMessage(xml);
        });
        setTimeout('updateMsg()',4000000);
}
```

（3）创建 MessageServlet 类，在 doPost()方法中获取请求参数，然后查询数据库中是否存在相同的信息，如果不存在则将当前的新消息保存到数据库中，最后返回 XML 文档结果，并将消息返回。代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        String user = request.getParameter("name");                //请求参数中的用户名
        String msg = request.getParameter("message");              //消息内容
        Message message = new Message();
        message.setUser(user);//用户名
        message.setMsg(msg);//消息内容
        message.setTime(Calendar.getInstance().getTimeInMillis());  //设置时间戳
        response.setContentType("text/xml ; charset=UTF-8");
        PrintWriter out = response.getWriter();
        if(!MessageDao.getInstance().isHasMsgByUserAndText(user, msg)){
                MessageDao.getInstance().saveMsg(message);          //保存信息
                int rows = MessageDao.getInstance().selectMsgCount();//查询数据行数
                if(rows>10){                                        //如果数据库表的数据超过 10 条，进行删除处理
                        MessageDao.getInstance().deleteMsg(rows);
                }
        }
        out.println("<?xml version='1.0' encoding='UTF-8'?>");
        out.println("<response>");
        out.println("<status>1</status>");
        out.println("<time>"+message.getTime()+"</time>");
        out.println("<message>");
        out.println("<user>"+message.getUser()+"</user>");
        out.println("<text>"+message.getMsg()+"</text>");
        out.println("</message>");
        out.println("</response>");
}
```

（4）创建 MessageDao 类，编写保存信息的方法、查询信息的方法、删除信息的方法和查询消息总数的方法。这些方法都比较简单，具体代码参见配书光盘。

（5）在 index.jsp 中，编写 addMessage()方法，用于解析$.post()的回调函数返回的 XML 文档结果（在这个 XML 结果中包含了服务器返回的用户发送的消息内容），然后将解析的消息内容展示在聊天窗口中。代码如下：

```
function addMessage(xml){
        if($('status',xml).text()=="2") return;              //如果状态为 2，则终止
        timestamp = $("time",xml).text();                     //更新时间戳
        $('message',xml).each(function(){
                var user = $('user',this).text();             //发布者
                var content = $('text',this).text();          //内容
                var htmlStr = "<strong>"+user+"</strong>： "+content+"<br />";
                $('#messagewindow').prepend(htmlStr);
        });
}
```

心法领悟 338：jQuery 获取 XML 文档节点的值。

通过 jQuery 解析 XML 文档非常简单，如果在回调函数中指定服务器返回的数据类型为 XML，那么就可以直接通过$.(element,xml).text()获取 XML 节点的值。其中，参数 element 为节点的名称；xml 为 XML 文档对象。例如，在 XML 文档中存在一个<message>节点，通过$('message',xml).text()即可获取<message>节点的值。有关 jQuery 解析 XML 文档的详细说明参见 jQuery 官方网站提供的 API 文档。

第 13 章

Dojo 框架

- Dojo 的常用 Widget
- Dojo 的基本应用
- Dojo 对 Ajax 的支持

13.1　Dojo 的常用 Widget

实例 339	实现网页按钮 光盘位置：光盘\MR\13\339	高级 实用指数：★★★★

实例说明

作为一款面向对象的 JavaScript 工具库，Dojo 提供了强大而实用的丰富功能。其代码被划分为逻辑单元，通常称之为模块。模块类似于 Java 中的包。除此之外，Dojo 还包含有简单的函数。本实例实现的是在页面中添加 Dojo 中封装的页面按钮，运行结果如图 13.1 所示。

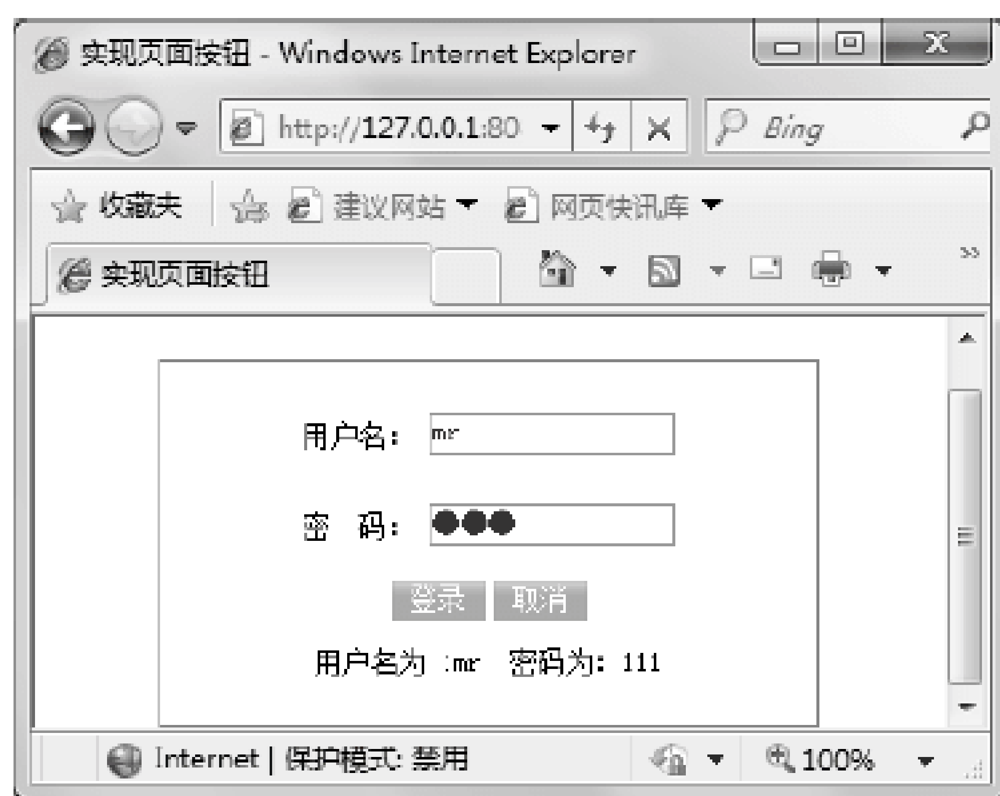


图 13.1　实现网页按钮

关键技术

要在项目中应用 Dojo 框架，首先要下载 Dojo 框架。在项目中引入 Dojo 的步骤如下：

（1）复制 Dojo 压缩文件夹下的 dojo.js 文件和 src 文件夹到 Web 应用的任意路径下。

（2）在页面中使用如下代码引入 Dojo 的 JavaScript 库。代码如下：

```
<script type="text/javascript" src="dojojs/dojo.js"></script>
```

（3）在页面中引入 Dojo 的 JavaScript 库还不够，还需要动态加载某些需要的函数和对象，例如，加载 dojo.widget 包下的所有函数和对象。代码如下：

```
<script type="text/javascript">
 dojo.require("dojo.widget.Button");
</script>
```

本实例应用了 Dojo 提供的 HTML 页面控件中的按钮控件。通过 Dojo 中的 HTML 控件，可以在页面中添加互动性很强而且很美观的元素。Dojo 中的按钮有 3 种类型，分别介绍如下。

- Button：普通按钮，该按钮不仅界面美观，而且没有提供额外的功能。
- DropDownButton：下拉菜单按钮，当单击该按钮时，将弹出下拉菜单。
- ComboButton：复合按钮，该按钮可以作为普通按钮使用，也可以弹出下拉菜单。

（1）在项目中定义 JavaScript 函数，实现在页面中导入 Dojo 中的按钮。具体代码如下：

```
<script type="text/javascript">
        dojo.require("dojo.widget.Button");                //导入 Dojo 框架中的 Button
</script>
```

（2）在页面中定义表格，应用 Dojo 中的按钮。具体代码如下：

```
<div align="center">
        <p>
                用户名：
        <input type="text" name="username" size=16 maxLength=20 />
        </p>
        <p>
                密  码：
                                <input type="password" name="password" size=16 maxLength=20 />
                        </p>
                </div>
        </td>
</tr>
<tr>
        <td height="37" align="center">
```

```
                <table>
                    <tr>
                    <td><button dojoType="Button" widgetId="btn" onclick="form1.submit()">登录</button></td>  //应用 Dojo 类的按钮
                        <td><button dojoType="Button" widgetId="btn1">取消</button></td>
                    </tr>
                </table>
            </td>
        </tr>
```

秘笈心法

心法领悟 339：Dojo 为什么会有“模块”和“包”这样的概念。

Dojo 中引入“模块”和“包”的概念，主要是为了满足应用程序只需要加载其所用到的内容的需要。充分利用模块化设计的优点，可以确保程序员交付最相关的功能代码，最大程度地减少代码的膨胀和消除及由此带来的不好的用户体验。



实例 340	实现网页对话框 光盘位置：光盘\MR\13\340	高级 实用指数：★★★★

实例说明

与 JavaScript 类似，Dojo 中也封装了一些对话框。当页面弹出对话框时，使用该对话框可以访问，页面的其他内容都会变为灰色。运行本实例，将显示“我的相册”中的内容；当用户单击“查看大图”按钮时，指定相片将以大图显示，如图 13.2 所示。



图 13.2　实现网页对话框

本实例实现在页面中显示对话框的原理为：在页面中定义<div>层，将该<div>层的 dojoType 属性设置为 Dialog；通过 bgColor 属性设置对话框的背景色；通过 bgOpacity 设置背景色的透明度；通过 toggle 属性设置关闭方式（通常为渐隐方式，即将 toggle 设置为 false）；通过 toggleDuration 设置对话框的消失时间。例如，本

实例中定义的对话框层。代码如下：

```
<table>
 <tr>
       <td align="center"><h3>树</h3></td>
</tr>
<tr><td><img src="2.bmp" width="400" height="400"></td></tr>
<tr><td align="center" ><input type="button" id=hider1 value="关闭" ></td></tr>
</table>
```

对话框还有 3 个常用方法。

- show()：设置对话框显示。
- setTimerNode()：设置对话框的倒计时容器，一旦将某个HTML节点设置成对话框的倒计时容器，该节点的原有内容将被覆盖。
- setCloseControl：设置对话框的关闭按钮。

（1）在页面中定义 JavaScript 函数，首先导入 dojo.widget。具体代码如下：

```
dojo.require("dojo.widget.*");
```

（2）本实例首先实现了显示小的相片，在每个相片下方都有一个“查看大图”按钮，当用户单击该按钮时，显示指定的对话框。关键代码如下：

```
<td>
        <button dojoType="Button" onclick="dlg.show()">查看大图</button>
</td>
<td>
       <button dojoType="Button" onclick="dlg1.show()">查看大图</button>
</td>
<td>
       <button dojoType="Button" onclick="dlg2.show()">查看大图</button>
</td>
```

（3）定义表示对话框的<div>层，控制在页面中显示的对话框。具体代码如下：

```
<div dojoType="Dialog" id="dialog1" bgColor="white" bgOpacity="0.5" toggle="fade" toggleDuration="250">
 <form onsubmit="return false;">
<table>
 <tr>
       <td align="center"><h3>树</h3></td>
</tr>
<tr><td><img src="2.bmp" width="400" height="400"></td></tr>                    //显示图片
<tr><td align="center" ><input type="button" id=hider1 value="关闭" ></td></tr>       //给出“关闭”按钮
</table>
</form>
</div>
```

（4）定义 JavaScript 函数，该函数在页面加载时执行。具体代码如下：

```
<script type="text/javascript">
       dojo.require("dojo.widget.*");
       var dlg,dlg1,dlg2,dlg3;
       function init(){
              dlg=dojo.widget.byId("dialog");                              //获取第一个对话框
              var btn=document.getElementById("hider");
              dlg.setCloseControl(btn);                                    //设置对话框的关闭控件
              dlg1=dojo.widget.byId("dialog1");
              var btn=document.getElementById("hider1");
              dlg1.setCloseControl(btn);
              dlg2=dojo.widget.byId("dialog2");
              var btn=document.getElementById("hider2");
              dlg2.setCloseControl(btn);
              dlg3=dojo.widget.byId("dialog3");
              var btn=document.getElementById("hider3");
              dlg3.setCloseControl(btn);
       }
       dojo.addOnLoad(init);
</script>
```

秘笈心法

心法领悟 340：Dojo 通用库。

Dojo 通用库提供了一些工具类、函数等，通过这些工具类、函数的辅助，可以更简单地操作 JavaScript 代码，并通过一些简单的方法来控制 HTML 元素、DOM 元素等。该通用库中通常包含以下几个包。

- dojo.*：该包中包含 Dojo 的一些基础对象和方法。
- dojo.lang.*：该包中包含一些工具函数，这些工具函数可以简化 JavaScript 操作。
- dojo.string.*：该包中包含一些操作字符串的程序。
- dojo.dom.*：该包中包含一些操作 DOM 元素的程序。
- dojo.style.*：该包中包含一些操作 CSS 样式的程序。

实例 341	实现日历功能 光盘位置：光盘\MR\13\341	高级 实用指数：★★★★

实例说明

在程序开发中，很多时候都需要添加日期。此时如果页面中给出日期提示框，用户选择起来将会非常方便。本实例将使用 Dojo 提供的日历功能，实现在页面中添加日历，运行结果如图 13.3 所示。



图 13.3　实现日历功能

关键技术

Dojo 提供了两种类型的日历，一种用于在文档中突出显示某一天，例如，告诉浏览者当天的日期；另一种则用于为浏览器提供日期选择。不管是哪一种日历，其语法都差不多，具体如下：

```
<div dojoType="dropdowndatepicker" name="date" widgetId = "test" value = "date" weekStartsOn= "5" displayWeeks = "5"
      startDate = "start" endDate = "end" lang = "area">
</div>
```

参数说明

❶ dojoType：设置日历的类型，可以选择参数 dropdowndatepicker 或 datepicker，前一个表示日历选择器，后一个表示普通日历。

❷ value：该参数用于初始化日历的当前时间，如果没有进行设置，将读取客户端的系统时间。

❸ weekStartsOn：设置每个星期的第一天是星期几，设置 5 则表示星期五为第一天。

❹ startDate：设置日历的开始时间。

❺ endDate：设置日历的结束时间。

❻ lang：设置日历的语言区域，默认读取客户端的系统语言。

（1）在页面中定义 JavaScript 函数，实现应用 Dojo 类库。具体代码如下：

```
<script type="text/javascript">
        dojo.require("dojo.widget.*");
</script>
```

（2）在页面中定义元素，为用户提供可添加的商品信息，以及日历信息。具体代码如下：

```
<tr>
        <td bgcolor="#f1f3f5">商品名称：</td>
        <td>
                <input type="text" name="name" />
        </td>
</tr>
<tr>
        <td bgcolor="#f1f3f5">商品数量：</td>
        <td>
                <input type="text" name="number" />
        </td>
</tr>
<tr>
        <td bgcolor="#f1f3f5">商品价格：</td>
        <td><input type="text" name="price"/></td>
</tr>
<tr>
        <td bgcolor="#f1f3f5">入库时间：</td>
        <td><div dojoType="dropdowndatepicker" name="date"></div></td>                //添加日历对话框
</tr>
<tr>
        <td colspan="2" align="center"><input type="submit" value="添加" /></td>
</tr>
```

秘笈心法

心法领悟 341：要将 dojo.js 与 src 放置在同一个目录下。

使用 Dojo 时，不仅需要将 dojo.js 复制到目标应用中，还需要将整个 src 文件夹复制到目标文件中，即一定要将 dojo.js 和 src 文件夹放置在相同的路径下。实际上，因为 Dojo 支持动态下载，因此使用哪个版本的 dojo.js 文件并不会导致页面彻底不能使用，即使引入了最小版本的 dojo.js，只要使用相应的动态下载，也一样可以使用 Dojo 的相应功能。

实例 342	实现网页的多页面 光盘位置：光盘\MR\13\342	中级 实用指数：★★★★

实例说明

实现一个页面的多页面效果，类似于 Windows 编程中的 Tab 效果，允许在一个页面内容纳更多的内容。Tab 页将窗口的内容分成几个部分，每个部分放入一个 Tab 页内。每次只能显示一个 Tab 页面的内容，一旦单击某个 Tab 标签，对应 Tab 页的内容就会显示出来。本实例将实现网页的多页面效果，运行结果如图 13.4 所示。



图 13.4　多页面

关键技术

Dojo 中的 Tab 页由两个 Widget 组成，一个是整个 Tab 容器，另一个是每个 Tab 页面。这两个类分别对应的 Dojo Widget 类如下。

- TabContainer：Tab 容器，每个容器对应一个 Tab 容器。TabContainer Widget 通常包含如下两个属性。
  - selectedChild：设置默认选择的 Tab 页。
  - doLayout：该属性设置 Tab 容器的高度是否随 Tab 页高度的变化而变化。
- ContentPane：Tab 页面，多个 Tab 页组成一个 Tab 容器。ContentPane Widget 包含如下几个属性。
  - href：指定该 Tab 页面内显示页面的 URL 地址。
  - label：指定该 Tab 页的标签。
  - handler：指定单击该 Tab 时触发的事件处理函数。

设计过程

在页面中定义<div>层，实现创建 Tab（标签）页。具体代码如下：

```
<div id="tabContainer" dojoType="TabContainer" doLayout="false" selectedChild="contact">   //定义 Tab 容器
<div id="brief" dojoType="ContentPane" label="公司简介">                                   //定义 Tab 标签页
<center> <h3>明日科技有限公司</h3></center>
<p>    吉林省明日科技有限公司是一家以计算机软件和数字出版为核心的高科技企业，
多年来始终致力于行业管理软件、数字化出版物等领域的实践，
目前已和多家国内上市企业形成产品合作关系，公司的多款软件产品应用于国内的多家企业集团。
明日科技拥有软件开发和项目实施方面的资深专家和学习型技术团队，利用积累的专业知识和经验，
已开发数百种专业技术图书和数十种数字化学习产品，成为国内 IT 信息服务领域的知名品牌。<br>
</p>
</div>
        <div id="contact" dojoType="ContentPane" label="联系我们">                        //定义 Tab 标签页
        <center><h3>联系我们</h3></center>
                  地  址：  xxx 街 xxx 号<br>
                  电  话：  123****<br>
                  邮  箱：  xxx@xxx<br>
                  联系人：  xxx<br>
        </div>
</div>
```

秘笈心法

心法领悟 342："mr".equals(username)与 username.equals("mr")。

这两条语句在语法上来说是相同的，但是在语义上稍有差别。username 是一个变量，既然是变量，就有可能是 null。如果值为 null，则调用任何方法都会失败。因此，使用后者时可能会报错，而前者不会。

13.2 Dojo 的基本应用

实例 343	鼠标单击事件处理 光盘位置：光盘\MR\13\343	高级 实用指数：★★★★

Dojo 的事件机制消除了 Internet Explorer 和 DOM 标准之间的事件冲突，允许开发者使用相同的代码来实现跨浏览器的事件监听。Dojo 不仅允许对 DOM 对象绑定事件监听函数，甚至可以对广义的 JavaScript 对象绑定事件监听器。本实例将在页面中添加鼠标单击事件处理，实现计算器的功能，运行结果如图 13.5 所示。

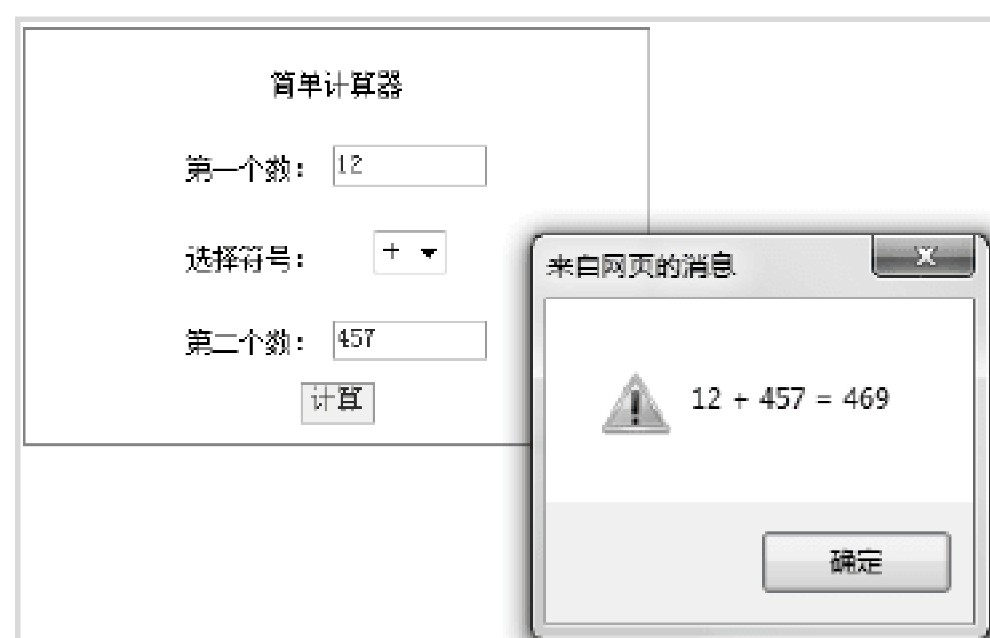


图 13.5　鼠标单击事件处理

关键技术

本实例实现为按钮添加单击事件，使用了 Dojo 提供的 dojo.event.connect()函数。该函数可以非常简单的方式为指定对象绑定事件监听函数。具体语法如下：

```
dojo.event.connect(handlerNode, "eventName",obj,handler)
```

参数说明

❶ handlerNode：绑定监听函数的 DOM 节点（HTML 元素），也可以是一个普通 JavaScript 对象。

❷ eventName：绑定对象的 JavaScript 函数。

❸ obj：该参数可以省略，此时相当于 obj 为 Window 对象，系统将从 Window 对象中搜索 handler 方法。

❹ handler：DOM 节点触发的方法。

（1）在页面中定义文本框，实现为用户提供添加数字的文本框。具体代码如下：

```
<p>简单计算器</p>
        <p>
                第一个数：
                <input type="text" id="one" size=10>                    //定义供用户添加内容的文本框
        </p>
        <p>
                选择符号：   
                <select name="sign">                                    //算术运算符
                   <option value="1">+</option>
                   <option value="2">-</option>
                   <option value="3">*</option>
                   <option value="4">/</option>
              </select>   
        </p>
        <p>
                第二个数：
                <input type="text" id="two" size=10>                    //定义供用户添加内容的文本框
</p>
```

（2）在页面中定义 JavaScript 函数，实现将用户输入的内容进行算术运算，并将运算结果显示给用户。具体代码如下：

```
<script type="text/javascript">
        var btn = document.getElementById("btn");                        //获取用户单击的对象
        function btnOnclick() {                                          //定义单击事件
        var sign=document.getElementById("sign").value;                  //获取用户选择的运算符
        var one=document.getElementById("one").value;                    //获取用户添加的内容
        var two=document.getElementById("two").value;
        var result;
                if(sign==1){                                             //如果用户选择了加法运算
                        result = parseFloat(one) + parseFloat(two);      //进行加法运算
                        alert((one+" + "+two+" = ")+result);
                }
                if(sign==2){                                             //如果用户选择了减法运算
                        result=one-two;
                        alert(one+" - "+two+" = "+result);
                }
```

```
        if(sign==3){                                    //如果用户选择了乘法运算
            result=one*two;
            alert(one+" * "+two+" = "+result);
        }
        if(sign==4){                                    //如果用户选择了除法运算
            result=one/two;
            alert(one+" / "+two+" = "+result);
        }
    }
    dojo.event.connect(btn, "onclick", "btnOnclick");
</script>
```

秘笈心法

心法领悟 343：在 JavaScript 中进行加法运算。

在 JavaScript 中用 var 运算符定义变量，该变量是弱类型，也就是说可以将变量初始化为任意的值，所以当用“+”号相加时，系统会把两个数据当作字符串相加；如果要做加法，则需要给数据规定类型。以给定 Int 型为例，代码如下：

```
var a=1,b=2;
var c;
c=parseInt(a)+parseInt(b);
```



实例 344 访问被监听方法的参数

光盘位置：光盘\MR\13\344

中级

实用指数：★★★★

实例说明

实例 343 介绍了如何使用 connect()函数实现监听按钮单击事件，实际上，connect()函数的参数并不一定是 DOM 节点，也可以是一个 JavaScript 对象。本实例实现访问被监听方法的参数，运行结果如图 13.6 所示。

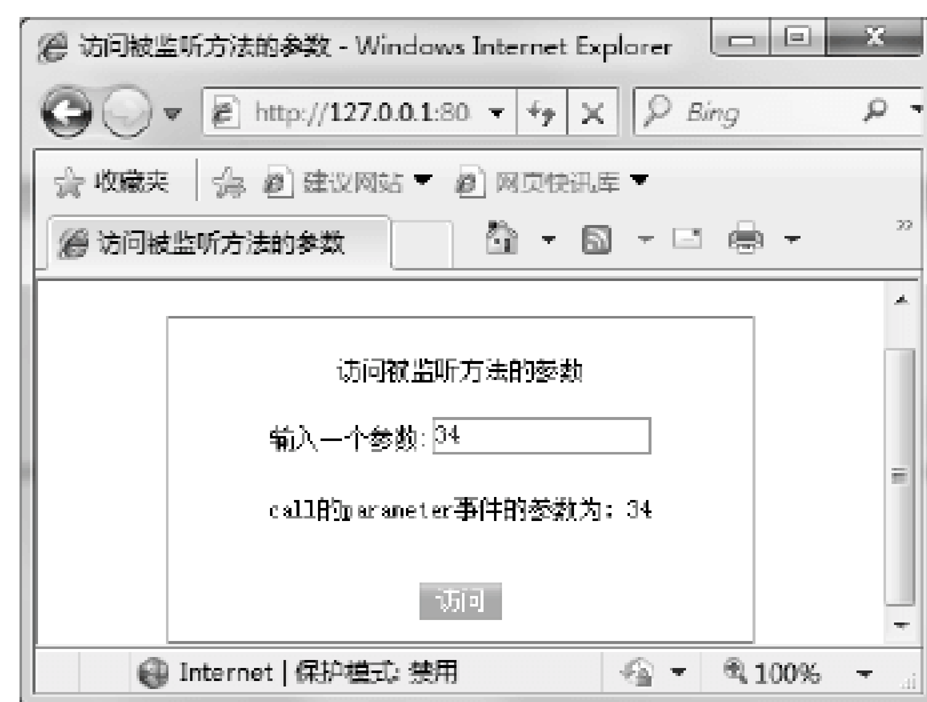


图 13.6　访问被监听方法的参数

关键技术

本实例实现了 connect()函数，有关该函数的语法可参见实例 343。

（1）在页面中定义表格，显示内容。具体代码如下：

```
<table width="274" height="123" border="0" align="center"
    cellpadding="0" cellspacing="0">
    <tr>
        <td height="35">
            <br>
            <div align="center">
                <p>
                访问被监听方法的参数<br><br/>
                    输入一个参数:<input type="text" name="t1" size=16 maxLength=20 />
                </p>
                <p>
                    <div id="message"></div>
                </p>
            </div>
        </td>
    </tr>
    <tr>
        <td height="37" align="center">
            <table>
                <tr>
                    <td><button dojoType="Button" widgetId="btn" onClick="monitor();">访问</button></td>
```

```
                        </tr>
                    </table>
                </td>
            </tr>
</table>
```

（2）在页面中编写 JavaScript 函数，实现当用户单击“访问”按钮时，显示用户输入参数。具体代码如下：

```
<script type="text/javascript">
        dojo.require("dojo.widget.*");                                                          //导入包
            var call={
                parameter : function(str){
                    dojo.byId("message").innerHTML="call 的 parameter 事件的参数为："+str;        //在页面中追加内容
                }
            };
            dojo.event.connect(call,"parameter");
            function monitor(){
                var str = document.getElementById("t1").value;
                call.parameter(str);
            }
</script>
```

秘笈心法

心法领悟 344：div 中的 style 属性。

在 div 中有一个 style 属性，通过设置其子属性，可以让 div 更加美观、整洁。style 中常用的属性有：height，设置 div 的高度；width，设置 div 的宽度。此外，还有许多效果和样式，在此不一一说明。用法如下：

```
<div style="width:200px;height:300px;background-color:#999999">
```



实例 345　页面 HTML 元素的任意移动

光盘位置：光盘\MR\13\345

高级

实用指数：★★★★☆

实例说明

Dojo 提供了一种拖动功能，即允许用户拖动页面元素，从而改变页面中 DOM 元素的位置。本实例应用 Dojo 的拖动技术实现了由 Web 程序完成的拼图，运行结果如图 13.7 所示。



图 13.7　页面 HTML 元素的任意移动

关键技术

本实例使用了 Dojo 拖动功能中的自由拖动方式，自由拖动是指用户可以自由拖动 HTML 元素，当释放鼠标后，HTML 元素将停留在鼠标释放的位置。

HTML 元素的自由拖动可通过 dojo.dnd.HtmlDragMoveSource 类来实现，该类的构造方法可接收一个 DOM 节点，该节点表示自由拖动的 HTML 元素。

（1）本实例为用户提供了可移动的拼图，用户还可以通过单击“查看拼图原图”按钮实现查看拼图的原图。在页面中定义图片以及可显示原图的<div>层，具体代码如下：

```
<br>
        <img id="photo1" src="image/7.jpg" width="150" height="150">
        <img id="photo2" src="image/8.jpg" width="150" height="150">
        <img id="photo3" src="image/3.jpg" width="150" height="150">
        <img id="photo4" src="image/4.jpg" width="150" height="150">
        <img id="photo5" src="image/9.jpg" width="150" height="150">
```

```
        <img id="photo6" src="image/6.jpg" width="150" height="150">
        <img id="photo7" src="image/1.jpg" width="150" height="150">
        <img id="photo8" src="image/2.jpg" width="150" height="150">
        <img id="photo9" src="image/5.jpg" width="150" height="150">
    <div dojoType="Dialog" id="dialog" bgColor="white" bgOpacity="0.5" toggle="fade" toggleDuration="250">   //定义显示大图的<div>层
        <form onsubmit="return false;">
            <table>
                <tr><td><img src="image/0.bmp" width="400" height="400"></td></tr>              //页面显示图片
                <tr><td align="center" ><input type="button" id=hider value="关闭" ></td></tr>      //在页面中添加“关闭”按钮
            </table>
        </form>
</div>
```

（2）在页面中定义 JavaScript 函数，实现为页面中的图片添加自由拖动动作，并为“查看拼图原图”按钮绑定事件。具体代码如下：

```
<script type="text/javascript">
            dojo.require("dojo.dnd.HtmlDragMove");                          //导入所需类
            dojo.require("dojo.event.*");
            dojo.require("dojo.widget.*");
            //该函数的参数在页面加载时执行
            dojo.addOnLoad(function(){
                new dojo.dnd.HtmlDragMoveSource(dojo.byId("photo1"));       //将 HTML 元素 photo1 转为可自由移动的元素
                new dojo.dnd.HtmlDragMoveSource(dojo.byId("photo8"));
                new dojo.dnd.HtmlDragMoveSource(dojo.byId("photo3"));
                new dojo.dnd.HtmlDragMoveSource(dojo.byId("photo9"));
                new dojo.dnd.HtmlDragMoveSource(dojo.byId("photo2"));
                new dojo.dnd.HtmlDragMoveSource(dojo.byId("photo6"));
                new dojo.dnd.HtmlDragMoveSource(dojo.byId("photo7"));
                new dojo.dnd.HtmlDragMoveSource(dojo.byId("photo5"));
                new dojo.dnd.HtmlDragMoveSource(dojo.byId("photo4"));
            });
        var dlg;
        function init(){
            dlg=dojo.widget.byId("dialog");                                 //获取表单元素内容
            var btn=document.getElementById("hider");
            dlg.setCloseControl(btn);                                       //关闭按钮
        }
        dojo.addOnLoad(init);
</script>
```

秘笈心法

心法领悟 345：阻止表单的自动提交。

在进行 Web 开发中，常常会遇到一按 Enter 键表单就自动提交的问题。对此可以通过设置 form 中的 onSubmit 属性来解决。

```
<form action="" onsubmit="return false;">
```

onsubmit="return false;"表示禁止提交表单。可以写一个 Script 验证来判断是否让表单提交，代码如下：

```
<form action="" onsubmit="return check();">
<script type="text/javascript">
        function check(){
            //验证表单如果不满足条件
             return false;
        }
    </script>
```



实例 346	页面元素的相对移动 光盘位置：光盘\MR\13\346	高级 实用指数：★★★★

实例说明

页面元素的相对移动是指只能把元素移动到指定的表单位置，而不能随意地进行移动。要实现这样

的效果可以使用 HtmlDragSource 类。本实例实现的是将页面中提供的植物移动到指定的土地，而不能移动到其他的位置，如图 13.8 所示。

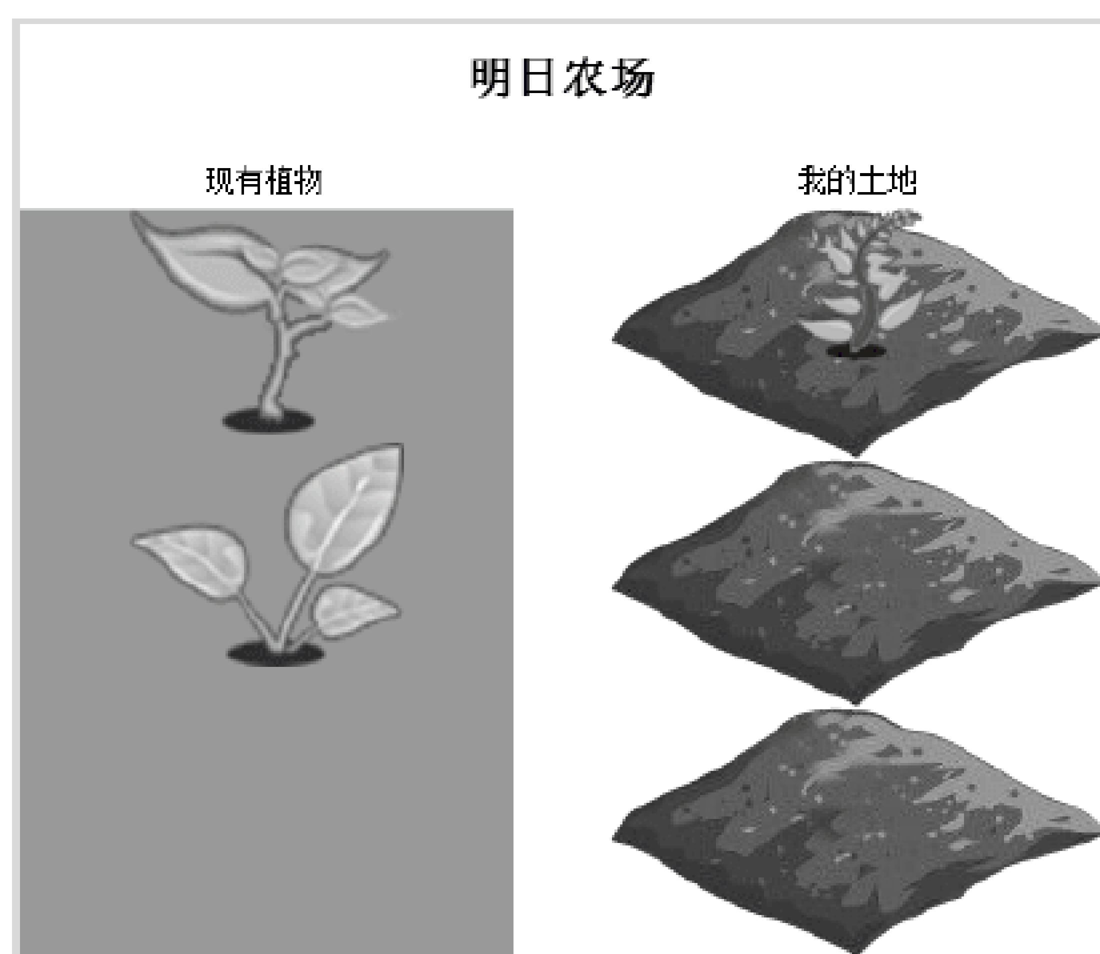


图 13.8　页面元素的相对移动

关键技术

利用 HtmlDragSource 类可以实现 HTML 元素的相对移动。HtmlDragSource 类表示拖动的“坐标”节点，构造方法语法如下：

```
dojo.dnd.HtmlDropTarget(targetNode,StrArray)
```

参数说明

❶ targetNode：“坐标”节点。

❷ StrArray：字符串数组，数组中的每个元素都将作为“坐标”的名称。

```
dojo.dnd.HtmlDraySource(sourceNode,name)
```

参数说明

❶ sourceNode：指定可拖动的 HTML 元素。

❷ name：指定元素的“坐标”节点名称。

设计过程

（1）在页面中定义代码，显示植物图片和土地图片。具体代码如下：

```
<div id="farm"
        style="width: 200px; height: 300px; background-color: #999999">
        <div id="f1">
                <img src="image/1.png" height="60">                              //在页面中显示植物图片
                </div>
                <div id="f2">
                        <img src="image/2.png" height="90">
                </div>
                <div id="f3">
                        <img src="image/3.png" height="90">
                </div>
        </div>
</td>
<td width="30"></td>
<td>
        <div id="glebe"
                style="width: 200px; height: 300px; background-image: url(image/4.png)">    //在页面中显示土地图片
        </div>
</td>
```

（2）在页面中定义 JavaScript 函数，实现控制页面元素的相对移动。具体代码如下：

```
<script type="text/javascript">
        dojo.require("dojo.dnd.*");                                    //加载 dojo.dnd.*模块
        dojo.require("dojo.event.*");
        dojo.addOnLoad(function(){                                     //加载页面时调用函数
                var case1=dojo.byId("farm");                           //获取表单元素
                var case2=dojo.byId("glebe");
                new dojo.dnd.HtmlDropTarget(case1,"case2");            //定义可拖动的位置
                new dojo.dnd.HtmlDropTarget(case2,"case1");
                var list1=case1.getElementsByTagName("div");
                for(var i=0;i<list1.length;i++){
                        new dojo.dnd.HtmlDragSource(list1[i],"case1");
                }
        });
</script>
```


心法领悟 346：防止页面乱码。

"request.setCharacterEncoding("gbk")";的作用是把从 request 中取得的值进行重新编码，从而有效地避免乱码现象。



实例 347 带手柄的移动

光盘位置：光盘\MR\13\347

高级

实用指数：★★★★

实例说明

手柄的移动，指如果想移动某个 HTML 元素，必须用鼠标按在手柄上。要实现带手柄的移动，可以通过 setDragHandle()函数来完成。本实例实现在页面中添加图片，用户可通过图片手柄进行移动，如图 13.9 所示。

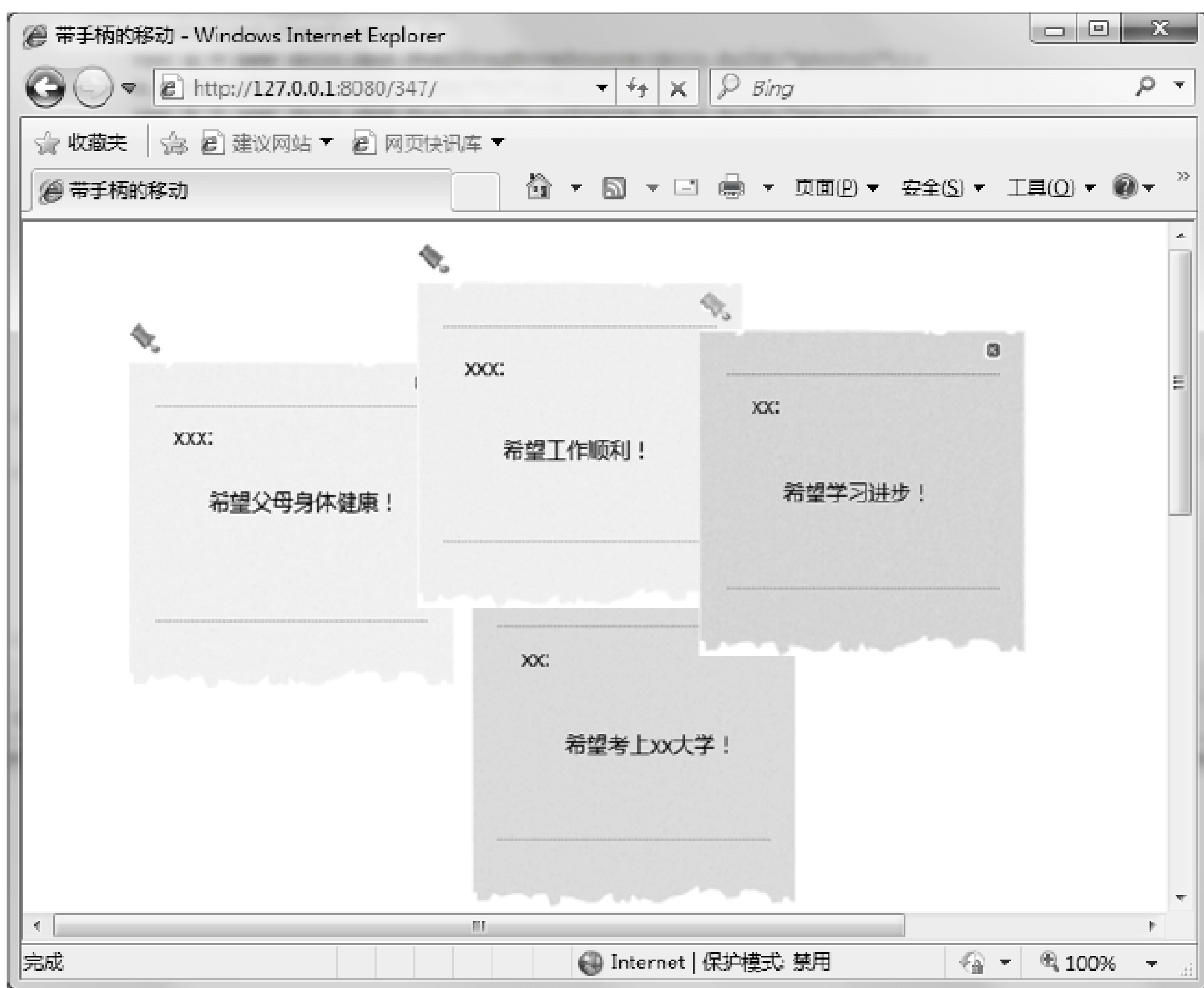


图 13.9　带手柄的移动

关键技术

HtmlDragSource、HtmlDragMoveSource 和 HtmlDragCopySource 实例都有 setDragHandle()方法，该方法可以实现为移动设置手柄。具体语法如下：

```
setDragHandle(node)
```

参数说明

node：调用该方法的元素的“手柄”。调用 setDragHandle()方法的元素必须是一个可移动的 HTML 元素。

（1）在页面中定义表格，添加并显示图片。具体代码如下：

```
<table width="890" height="1003" border="0" align="center" background="">
  <tr>
    <td colspan="4"> </td>
  </tr>
  <tr>
    <td width="263" height="511">
      <div id="photo1">            //在页面中定义<div>层显示图片内容
        <div id="t1">
              <img src="image/11.png" width="20" height="20">
      </div>
              <img src="image/1.png" width="200" height="200">
      </div>
      </td>
    <td width="188">
```

```
        <div id="photo3">
        <div id="t3">
                <img src="image/33.png" width="20" height="20">
        </div>
                <img src="image/3.png" width="200" height="200">
        </div>
        </td>
      <td width="221">
          <div id="photo2">
          <div id="t2">
                <img src="image/22.png" width="20" height="20">
        </div>
                <img src="image/2.png" width="200" height="200">
        </div>
        </td>
      <td width="200">
          <div id="photo4">
                <div id="t4">
                        <img src="image/44.png" width="20" height="20">
                </div>
                        <img src="image/4.png" width="200" height="200">
                </div>
        </td>
    </tr>
    <tr>
      <td colspan="4"> </td>
    </tr>
</table>
```

（2）在页面中定义 JavaScript 函数，实现将页面中的图片完成带手柄的移动。具体代码如下：

```
<script type="text/javascript">
        dojo.require("dojo.dnd.*");                        //在页面中应用相应模块
        dojo.require("dojo.dnd.HtmlDragMove");
        dojo.require("dojo.event.*");
        dojo.addOnLoad(function() {
                var a = new dojo.dnd.HtmlDragMoveSource(dojo.byId("photo1"));
                a.setDragHandle(dojo.byId("t1"));          //定义可移动手柄
                var b = new dojo.dnd.HtmlDragMoveSource(dojo.byId("photo2"));
                b.setDragHandle(dojo.byId("t2"));
                var c = new dojo.dnd.HtmlDragMoveSource(dojo.byId("photo3"));
                c.setDragHandle(dojo.byId("t3"));
                var d = new dojo.dnd.HtmlDragMoveSource(dojo.byId("photo4"));
                d.setDragHandle(dojo.byId("t4"));
        });
</script>
```

秘笈心法

心法领悟 347：datepicker 组件。

这也是 Dojo 提供的一种创建日历的组件，其用法和 dropdowndatepicker 一样。代码如下：

```
<div dojoType="datepicker" ></div>
```

13.3　Dojo 对 Ajax 的支持



实例 348　基本请求的发送

光盘位置：光盘\MR\13\348

高级

实用指数：★★★

Dojo 的 Ajax 能力主要依赖于 Dojo 包结构下的函数和类。Dojo 的 dojo.io.bind()函数提供了强大的 Ajax 处理能力，使用该函数只需指定请求的 URL，并发送指定的请求参数、指定特定的回调函数，Ajax 交互中的其他事

件都由 Dojo 完成。本实例将应用 Dojo 实现基本请求的发送，运行结果如图 13.10 所示。

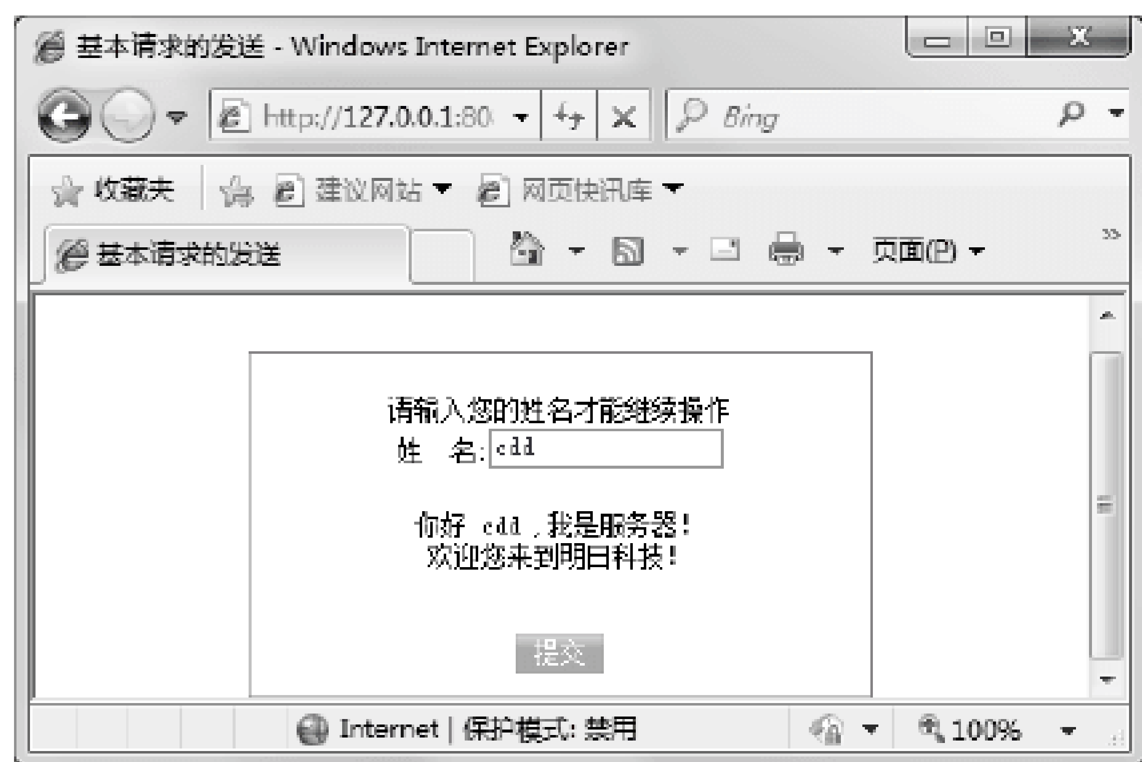


图 13.10　基本请求的发送

关键技术

dojo.io.bind()函数需要一个对象参数，该对象参数通常包含如下几个简单的属性。

- url：Ajax 请求发送的 URL 地址。
- method：发送请求的方式，GET 或 POST。
- handler：指定回调函数，当服务器响应完成时该函数自动启动。
- content：该属性是一个 JavaScript 对象，该对象由一组属性组成，由需要发送的请求参数组成。

设计过程

（1）在页面中定义表格，具体代码如下：

```
<table width="274" height="123" border="0" align="center"
        cellpadding="0" cellspacing="0">
        <tr>
                <td height="35">
                        <br>
                        <div align="center">
                                <p>
                                        请输入您的姓名才能继续操作<br>
                                        姓  名:<input type="text" name="name" size=16 maxLength=20 />    //定义输入信息的文本框
                                </p>
                                <p>
                                        <div id="login"></div>
                                </p>
                        </div>
                </td>
        </tr>
        <tr>
                <td height="37" align="center">
                        <table>
                                <tr>
                                        <td><button dojoType="Button" widgetId="btn" onClick="login();">提交</button></td>
                                </tr>
                        </table>
                </td>
        </tr>
</table>
```

（2）定义 JavaScript 代码，实现绑定请求地址、请求参数等。具体代码如下：

```
<script type="text/javascript">
        function login(){
                dojo.io.bind({
                        url:"server.jsp",                    //请求发送 URL 地址
                        method:"post",                       //发送请求方式
                        handler:Callback,
                        content:{name:dojo.byId("name").value}
                });
        }
        function Callback(info,data,obj){
                        dojo.byId("login").innerHTML=data;
        }
</script>
```

心法领悟 348：Dojo 处理字符串。

Dojo 还提供了很多处理字符串的方法，例如 dojo.string.capitalize()把每个单词的首字母大写，dojo.string.isBlank()判断字符串是否为空等，用法和 dojo.string.trim()基本相同。在此以 dojo.string.capitalize()为例。代码如下：

```
dojo.string.capitalize("abc de fg");
```

实例 349　请求队列的发送

光盘位置：光盘\MR\13\349

中级　实用指数：★★★★

实例说明

bind()函数用于发送一个请求，如果要发送请求队列，则可以使用 queueBind()函数。该函数会自动维护页面的请求队列，按时间依次发送请求。本实例实现的是使用 queueBind()函数发送请求队列，运行结果如图 13.11 所示。



图 13.11　请求队列的发送

关键技术

queueBind()函数的语法和 bind()函数完全相同。使用该函数时，即使某个时刻无法申请到新的 XMLHttpRequest 对象，queueBind()函数也会自动将请求加入请求队列，一旦获得有效的 XMLHttpRequest 对象，该请求即被发送。

（1）在页面中定义表格，在表格内显示图片和“购买”按钮，当用户单击该按钮时，调用相应的函数。具体代码如下：

```
<table width="1003" height="1046" background="0.JPG">
	<tr>
		<td width="430" height="334"></td>
		<td width="561"></td>
	</tr>
	<tr>
		<td height="108"></td>
		<td align="center">
			<p>
				产品名称：　显示器
				<br>
				产品单价：　800 元
				<br>
				购买数量：
				<input type="text" name="number" size=5 maxLength=20 value="1" />
				台
			</p>
			<button dojoType="Button" widgetId="btn"
				onClick="dlg.show();count();">
				购买
			</button>
		</td>
	</tr>
	<tr>
		<td></td>
		<td></td>
```

```
        </tr>
</table>
```

（2）在页面中定义 JavaScript 函数，通过 queueBind()函数发送请求队列。具体代码如下：

```
<script type="text/javascript">
        function count(){
            dojo.io.queueBind({
                url:"server.jsp",            //请求地址
                method:"post",
                handler:Callback,            //回调函数
                content:{number:dojo.byId("number").value}
            });
        }
        function Callback(info,data,obj){
                dojo.byId("show").innerHTML=data;
        }
</script>
```

（3）服务器页面 server.jsp 的代码如下：

```
<jsp:directive.page import="java.lang.*" />
<%
    int number=Integer.parseInt(request.getParameter("number"));
    String s="您所购买的产品<br/>名称为：显示器<br/>单价为：800 元<br/>购买数量："+number+"台<br/>本次购物您需要支付
"+number*800+" 元";
    out.println(s);
%>
```

秘笈心法

心法领悟 349：div 摆放位置的规范性。

div 属于 HTML 中的标签，只要是在 HTML 中写入，任何位置都是可以的。但是如果 div 里包含<form>，那么这个 div 就要写到<body>中。出于代码的规范性，建议大家把 div 写到<body>里。

实例 350	对象的字符串化 光盘位置：光盘\MR\13\350	中级 实用指数：★★★★

实例说明

Dojo 中提供了处理字符串的函数，这些函数都放在 dojo.string 包下。本实例将使用 Dojo 的字符串处理技术来实现将用户输入的内容进行去掉前后空格处理，运行结果如图 13.12 所示。

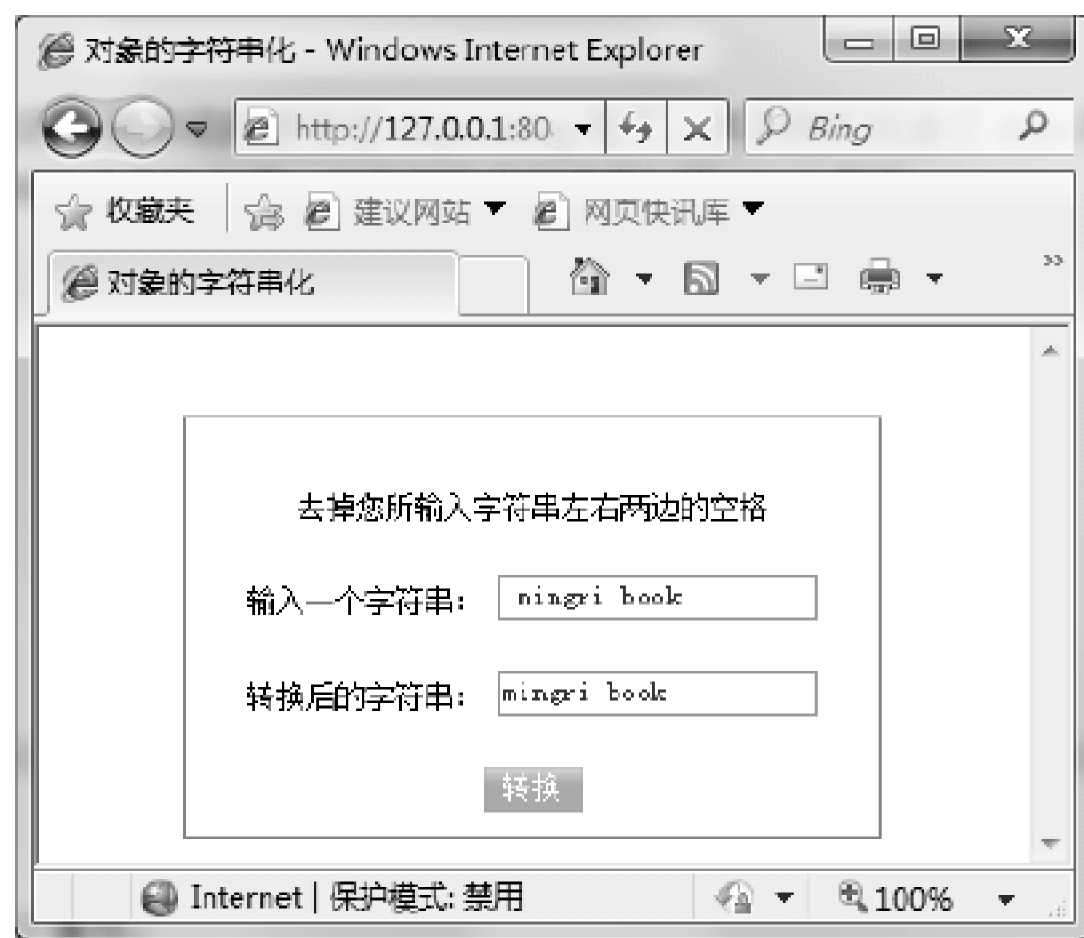


图 13.12 对象的字符串化

Dojo 中的字符串相关函数被放在 dojo.string 下，其中主要函数介绍如下。

- dojo.string.caplitalize(string str)：该函数将 str 中每个单词的首字母大写，Dojo 以空格为单词的分隔符。
- dojo.string.endsWith(string str,string end,Boolean igoreCase)：该函数判断 str 是否以 end 结尾，参数 igoreCase 用于指定是否忽略大小写。
- dojo.string.endcodeAscii(string str)：将参数字符串转换为 ASCII 格式的字符串。
- dojo.string.isBlanks(string str)：判断参数字符串是否为一个空串。
- dojo.string.padLeft(string str,integer len,string c)：在字符串的左边添加 c 字符串，添加后得到长度为 len 的字符串。

- dojo.string.padRight(string str,integer len,string c)：在字符串的右边添加 c 字符串，添加后得到长度为 len 的字符串。
- dojo.string.startsWith(string str,string start,boolean ignoreCase)：判断 str 是否以 start 开头，参数 ignoreCase 用于指定是否忽略大小写。
- dojo.string.trim(string str,integer wh)：去掉字符串前后的空格。参数 wh 为可选参数，如果没有指定该参数，将去掉字符串前后的空格；如果 wh 大于 0，则去掉左边的空格；如果 wh 小于 0，则去掉右边的空格。
- dojo.string.trimStart(string str)：去掉字符串 str 开头的空格。
- dojo.string.trimEnd(string str)：去掉字符串 str 后面的空格。

设计过程

（1）在项目的 index.jsp 页面中，定义表单文本框，供用户添加要进行处理的字符串。关键代码如下：

```
<table width="284" height="141" border="0" align="center"
        cellpadding="0" cellspacing="1" bgcolor="#6685C5">
        <tr>
                <td width="402" bgcolor="#FFFFFF">
                        <table width="274" height="170" border="0" align="center"
                                cellpadding="0" cellspacing="0">
                                <tr>
                                        <td height="133">
                                                <br>
                                                <div align="center">
                                                <p>去掉您所输入字符串左右两边的空格</p>
                                                        <p>
                                                                输入一个字符串：
                                                                <input type="text" name="t1">     //定义为用户提供处理的文本框
                                                        </p>
                                                        <p>
                                                                转换后的字符串：
                                                                <input type="text" name="t2">     //显示结果的文本框
                                                        </p>
                                                </div>
                                        </td>
                                </tr>
                                <tr>
                                <td height="37" align="center">
                                <button dojoType="Button" widgetId="btn" onclick="testString();">转换</button>        //为用户提供“转换”按钮
                                        </td>
                                </tr>
                        </table>
                </td>
        </tr>
</table>
```

（2）在页面中定义 Dojo 代码，实现对字符串前后空格进行去除。具体代码如下：

```
<script type="text/javascript" src="dojojs/dojo.js"></script>
<script type="text/javascript">
        dojo.require("dojo.widget.*");
        function testString() {
        var t1=document.getElementById("t1").value;                    //获取页面中表单文本域值
        document.getElementById("t2").value=dojo.string.trim(t1);    //将文本框内容去除空格显示
        }
</script>
```

心法领悟 350：onClick 触发两个事件。

在一个 onClick 中，可以同时触发两个事件。代码如下：

```
onClick="function1();function2();"
```



实例 351 表单请求发送

光盘位置：光盘\MR\13\351

高级

实用指数：★★★★

实例说明

本实例实现了应用 Dojo 整合 Ajax。与实例 348 功能类似，将表单内容以一个新的<div>层显示，运行结果如图 13.13 所示。

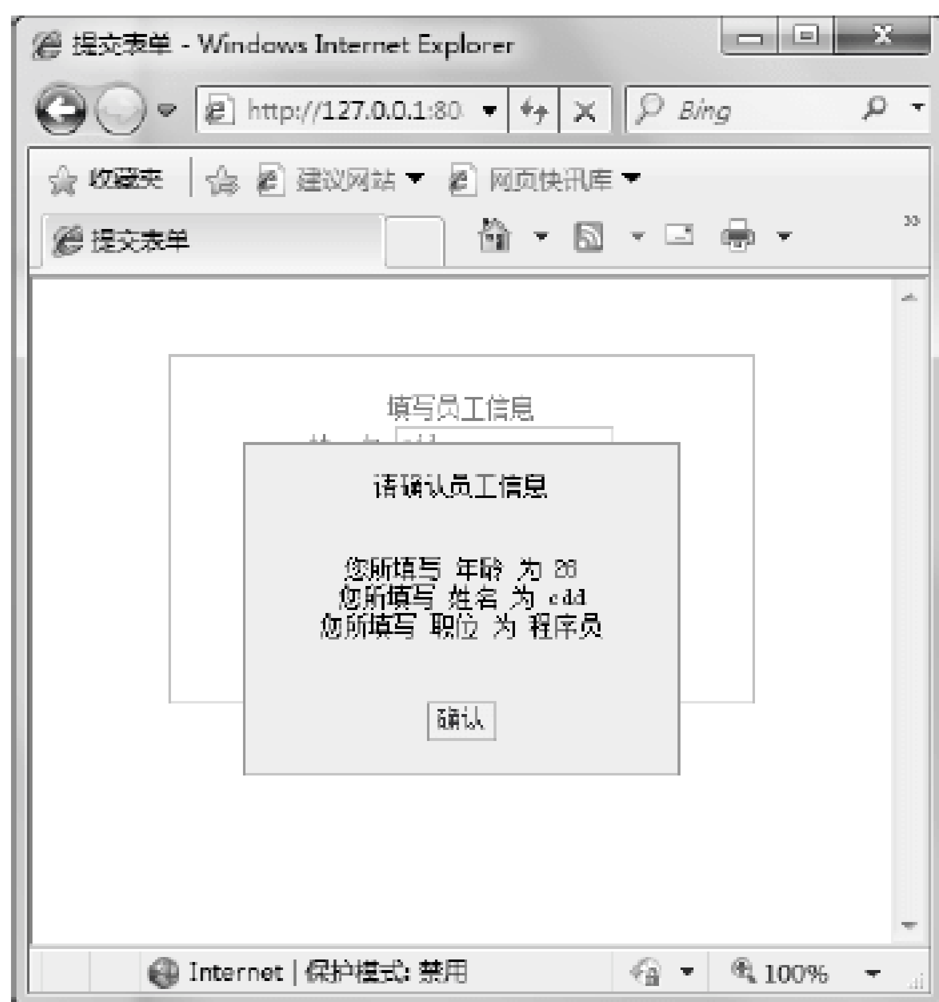


图 13.13　表单请求发送

关键技术

本实例应用 Dojo 处理 Ajax 请求，实现 dojo.io.bind()函数，有关该函数的具体语法可参见实例 348。

（1）在项目的 index.jsp 页面中，定义表格，为用户提供可填写的员工信息。具体代码如下：

```
<table width="284" height="50" border="0" align="center" cellpadding="0" cellspacing="1" bgcolor="#6685C5">
    <tr>
        <td width="402" bgcolor="#FFFFFF">
            <table width="274" height="123" border="0" align="center"
                cellpadding="0" cellspacing="0">
                <tr>
                    <td height="35">
                        <br>
                        <div align="center">
                            <p>
                                填写员工信息<br>
                                姓  名:<input type="text" name="姓名" size=16 maxLength=20 />
                                年  龄:<input type="text" name="年龄" size=16 maxLength=20 />
                                职  位:<input type="text" name="职位" size=16 maxLength=20 />
                            </p>
                            <p>
                            </p>
                        </div>
                    </td>
                </tr>
                <tr>
                <td height="37" align="center">
                <table>
                    <tr>
                     <td>
                        <button dojoType="Button" widgetId="btn" onClick="dlg.show();logins();">提交</button>
                        </td>
                        </tr>
```

```
                </table>
            </td>
</tr>
</table>
```

（2）定义<div>层，用于显示提交信息。具体代码如下：

```
<div dojoType="Dialog" id="dialog1" bgColor="white" bgOpacity="0.5" toggle="fade" toggleDuration="250">
        <form onsubmit="return false;">
            <table height="150" width="200">
                <tr>
                    <td align="center">请确认员工信息</td>
                </tr>
                <tr>
                    <td align="center">
                        <div id="login"></div>
                    </td>
                </tr>
                <tr>
                    <td align="center">
                        <input type="button" id="hider1" value="确认">
                    </td>
                </tr>
            </table>
        </form>
</div>
```

（3）定义 JavaScript 函数，控制请求处理。具体代码如下：

```
<script type="text/javascript">
    function logins(){
        dojo.io.bind({
            url:"server.jsp",                  //请求地址
            method:"post",                     //请求方式
            handler:Callback,                  //回调函数
            formNode:dojo.byId("form1")
        });
    }
    function Callback(info,data,obj){
        dojo.byId("login").innerHTML=data;
    }
</script>
```

（4）在 server.jsp 页面中，将获取的表单内容显示在页面中。具体代码如下：

```
<%
    request.setCharacterEncoding("gbk");                          //设置请求编码
    Enumeration enu=request.getParameterNames();                  //获取页面提交的请求参数
    String s="";
    while(enu.hasMoreElements()){                                 //循环遍历请求参数
        String ename=(String)enu.nextElement();                   //获取指定的参数内容
        s=s+"您所填写 "+ename+" 为 "+request.getParameter(ename)+"<br>";   //将获取的参数显示在页面中
    }
    out.println(s);
 %>
```

■ 秘笈心法

心法领悟 351：在 JavaScript 中每条语句的结尾处并不要求必须是“;”。

在 JavaScript 中每条语句的结尾处并不要求必须是“;”，也就是说可以加分号，也可以不加分号。如果语句结束时没有分号，JavaScript 会自动把该行代码的结尾作为语句的结尾；但是为了养成好的编程习惯，最好还是在结尾处加上分号，这样可以保证每行代码的准确性。

第5篇

流行框架篇

- ⏭ 第 14 章　Struts2 框架应用
- ⏭ 第 15 章　Struts2 框架标签应用
- ⏭ 第 16 章　Hibernate 框架基础
- ⏭ 第 17 章　Hibernate 高级话题
- ⏭ 第 18 章　Spring 框架基础
- ⏭ 第 19 章　Spring 的 Web MVC 框架

第14章

Struts2 框架应用

- Struts2 的基本配置与零配置
- Struts2 数据校验与拦截器
- 文件上传与下载
- Struts2 对 Ajax 的支持

14.1　Struts2 的基本配置与零配置

实例 352	成绩统计器 光盘位置：光盘\MR\14\352	中级 实用指数：★★☆

实例说明

在教学系统中经常设有对学生成绩进行统计的模块，本实例将实现对学生成绩的统计，运行结果如图 14.1 所示。

学生成绩统计

编号	姓名	性别	年龄	班级	成绩
1	张三	男	12	20805班	98分
2	李明	男	13	20805班	87分
3	王勃	男	12	20805班	68分
4	刘丽	女	11	20805班	76分
5	徐明	男	11	20805班	52分
6	叶硕	女	12	20805班	93分
7	王朝阳	男	12	20805班	90分
合计：					564分

图 14.1　统计学生成绩

关键技术

本实例利用聚集函数 SUM 对学生的成绩进行汇总。

SUM 聚集函数主要用于返回表达式中所有值的和，或只返回 DISTINCT 值。SUM 聚集函数只能用于数据类型是数字的列，null 值将被忽略。语法如下：

```
SUM ( [ALL|DISTINCT] expression )
```

参数说明

❶ ALL：对所有的值进行聚集函数运算。ALL 是默认设置。

❷ DISTINCT：指定 SUM 返回唯一值的和。

❸ expression：是常量、列或函数，或者是算术、按位与字符串等运算符的任意组合。如果 expression 是精确数字或近似数字数据类型分类（bit 数据类型除外）的表达式，则不允许使用聚集函数和子查询。

（1）创建 ShowALLDAO.java 类，在其中编写用于查询数据库字段的 ShowAll()方法以及用于计算总成绩的 sum()方法。具体代码如下：

```
public List<Student> ShowAll()
    {
        List<Student> list=new ArrayList<Student>();
        Connection con=DBUtil.con;                                          //得到一个数据库连接
        try{
            String sql="select id,name,sex,age,grade,classname from tb_student";    //查询数据库中字段
            PreparedStatement ps=con.prepareStatement(sql);                 //执行查询语句
            ResultSet rs=ps.executeQuery();                                 //得到查询结果
            while(rs.next())
```

```
            {
              Student st=new Student();
              st.setId(rs.getInt("id"));
              st.setName(rs.getString("name"));
              st.setSex(rs.getString("sex"));
              st.setAge(rs.getString("age"));
              st.setGrade(rs.getInt("grade"));
              st.setClassname(rs.getString("classname"));
              list.add(st);                                        //list 中存放每个 student 对象
            }
          } catch (SQLException e) {System.out.println(e.getMessage());
     }
          return list;
   }
public int ShowSum()
    {
        Connection con=DBUtil.con;
        int sum=0;
        try{
             String sql="select sum(grade) sum from tb_student";   //对数据库中的 grade 字段进行汇总
             PreparedStatement ps=con.prepareStatement(sql);       //执行 SQL 语句
             ResultSet rs=ps.executeQuery();                       //得到查询结果
             if(rs.next())
             {
              sum=rs.getInt("sum");
             }
          } catch (SQLException e) {System.out.println(e.getMessage());
              }
          return sum;
   }
```

（2）通过 struts.xml 中的配置进行转发，然后在 JSP 页面上进行显示。具体代码如下：

```
<table width="392" border="1">
        <s:iterator value="list">                                   //struts 标签迭代显示查询的数据
          <tr>
            <td width="54"><div align="center"><s:property value="id"/></div></td>
            <td width="58"><div align="center"><s:property value="name"/></div></td>
            <td width="52"><div align="center"><s:property value="sex"/></div></td>
            <td width="44"><div align="center"><s:property value="age"/></div></td>
            <td width="91"><div align="center"><s:property value="classname"/></div></td>
            <td width="53"><div align="center"><s:property value="grade"/>分</div></td>
          </tr>
        </s:iterator>
        <tr>
          <td colspan="5"><strong>  合计:</strong></td>
          <td><div align="center"><s:property value="sum"/>分</div></td>      //输出成绩汇总信息
        </tr>
      </table>
```

秘笈心法

心法领悟 352：注意 SUM 函数的使用。

SUM 函数只能用于数据类型是 int、smallint、tinyint、decimal、numeric、float、real、money 和 smallmoney 的字段，对于上述数据类型之外的数据则不予求和。

实例 353	成绩排序 光盘位置：光盘\MR\14\353	中级 实用指数：★★

实例说明

本实例实现的是将学生信息表中的成绩进行升序排序。在地址栏中输入 http://localhost:8080/353/showall.action，即可将学生信息表中的成绩升序排序，如图 14.2 所示。

学生成绩排序

编号	姓名	性别	年龄	班级	成绩
5	徐明	男	11	20805班	52分
3	王劲	男	12	20805班	68分
4	刘丽	女	11	20805班	76分
2	李明	男	13	20805班	87分
7	王朝阳	男	12	20805班	90分
6	叶硕	女	12	20805班	93分
1	张三	男	12	20805班	98分

图 14.2　成绩升序排序

■ 关键技术

要实现对数据进行升序排序查询，需在 SQL 语句中使用 ORDER BY 子句和 ASC 关键字，其实现方法是在 SQL 语句的查询语句中添加 ORDER BY GRADE ASC 子句。

ORDER BY 子句的作用是分类输出，并且根据表中包含的一列或多列表达式将输出的值按升序或降序排列。它并不改变数据库中行的顺序，只是简单地改变查询输出的值的显示顺序。语法如下：

```
SELECT ...
WHERE ...
ORDER BY expression [ASC|DESC],...
```

参数说明

❶ expression：一个表达式，通常是一个输出时用来分类的字段。

❷ [ASC/DESC]：可选项，代表升序或者降序。

❸ ,...：表示可以有多于一个的分类表达式，并且每个表达式都可以为升序或者降序。

排序表达式中可包括未出现在 SELECT 子句选择列表中的列名。如果在 SELECT 子句中使用了 DISTINCT 关键字，或查询语句中包含 UNION 运算符，则排序列必须包含在 SELECT 子句选择列表中。

■ 设计过程

（1）创建 ShowAllDao.java 类，在其中编写用于查询数据库字段，并且对 grade 字段进行升序排序的 ShowAll() 方法。具体代码如下：

```
public List<Student> ShowAll()
    {
            List<Student> list=new ArrayList<Student>();
            Connection con=DBUtil.con;
        try{
            String sql="select * from tb_student order by grade asc";        //查询所有字段并对成绩进行升序排序
            PreparedStatement ps=con.prepareStatement(sql);                  //执行查询语句
          ResultSet rs=ps.executeQuery();                                    //得到查询结果
          while(rs.next())
          {
            Student st=new Student();
            st.setId(rs.getInt("id"));
            st.setName(rs.getString("name"));
            st.setSex(rs.getString("sex"));
            st.setAge(rs.getString("age"));
            st.setGrade(rs.getInt("grade"));
            st.setClassname(rs.getString("classname"));
            list.add(st);                                                     //list 中存放每个 student 对象
          }
        } catch (SQLException e) {System.out.println(e.getMessage());
    }
            return list;
  }
```

（2）通过 struts.xml 中的配置进行转发，然后在 JSP 页面上进行显示。具体代码如下：

```
<table width="392" border="1">                                    //struts 标签迭代显示查询的数据
    <s:iterator value="list">
      <tr>
        <td width="54"><div align="center"><s:property value="id"/></div></td>
        <td width="58"><div align="center"><s:property value="name"/></div></td>
        <td width="52"><div align="center"><s:property value="sex"/></div></td>
        <td width="44"><div align="center"><s:property value="age"/></div></td>
        <td width="91"><div align="center"><s:property value="classname"/></div></td>
        <td width="53"><div align="center"><s:property value="grade"/>分</div></td>
      </tr>
    </s:iterator>
</table>
```

秘笈心法

心法领悟 353：数据库中数据的升/降序排序。

对数据库中的字段进行排序，默认情况下会按照某一列进行升序排序，也可以使用 DESC 进行降序排序。

实例 354 用户的直接登录

光盘位置：光盘\MR\14\354

中级

实用指数：★★☆

实例说明

本实例实现的是用户的直接登录。运行程序，在如图 14.3 所示页面中输入正确的用户名和密码，单击“登录”按钮，将转向登录成功页面，否则转向登录失败页面。



图 14.3　用户登录页面

关键技术

要实现用户的直接登录，首先要对用户输入的用户名和密码的正确性进行判断，这样就要用到 String 类中的 equals()方法。语法如下：

```
Boolean equals(String str)
```

参数说明

str：要作比较的字符串对象。

如果和 String 相等则为 true，否则为 false。

在项目中创建 LoginAction 类，在该类中定义成员变量 username、password 并提供 get()、set()方法，然后进行和特定字符串的比较。具体代码如下：

```
public class LoginAction extends ActionSupport{
        private String username;                              //定义成员变量 username
        private String password;                              //定义成员变量 password
        public String getUsername() {
              return username;
```

```
        }
        public void setUsername(String username) {
                this.username = username;
        }
        public String getPassword() {
                return password;
        }
        public void setPassword(String password) {
                this.password = password;
        }
                @Action("login")
        public String execute()
            {
                if(username.equals("mr")&&password.equals("mrsoft")){    //前台表单中的值与特定的字符串比较
                        return "success";                               //返回成功页面
                }
                else{
                        return "error";                                 //返回失败页面
                }
            }
}
```

秘笈心法

心法领悟 354：equal()方法和“==”的区别。

equals()方法比较的是对象的内容（区分字母的大小写格式），而“==”比较的是两个对象的内存地址。

实例 355　实现用户的中间退出

中级

光盘位置：光盘\MR\14\355

实用指数：★★★☆

本实例实现的是用户的中间退出。用户登录成功以后，会进入一个留言板的模块，如图 14.4 所示。单击该页面中的“安全退出”超链接，即可实现用户的退出，同时转向用户登录页面。



图 14.4　用户的中间退出

关键技术

要实现用户的中间退出，就要执行 Action 类中特定的方法，这是通过设置 href 属性值实现的。其属性值表示为 Action 名称后加“!”，然后加方法名。当用户退出以后将转向退出成功页面，这可以通过 Struts2 提供的“@”注解实现 Action 零配置，而不需要在 struts.xml 中进行相应的配置。

设计过程

在项目中创建类 LoginAction，在该类中使用“@”注解定义 Action 的资源。具体代码如下：

```
@Results({
        @Result(name="success",location="/success.jsp"),                    //配置登录成功页面
        @Result(name="error" ,location="/error.jsp"),                       //配置登录失败页面
        @Result(name="exit" ,location="/exit.jsp")                          //配置用户退出页面

})
public class LoginAction extends ActionSupport{
          private String username;                                          //定义成员变量 username
          private String password;                                          //定义成员变量 password
           public String getUsername() {
                  return username;
        }
        public void setUsername(String username) {
                  this.username = username;
        }
        public String getPassword() {
                  return password;
        }
        public void setPassword(String password) {
                  this.password = password;
        }
@Action("login")                                                            //配置登录方法
        public String execute(){
                   if(username.equals("mr")&&password.equals("mrsoft")){    //前台表单中的值与特定的字符串比较
                          return "success";                                 //返回成功页面
                   }
                   else{
                          return "error";                                   //返回失败页面
                   }
             }
@Action("ex")                                                               //配置退出方法
           public String exit(){
                 return "exit";
             }
```

秘笈心法

心法领悟 355：@Results 的作用和位置。

使用@Results 是定义一组 result 映射，配置的内容必须是在类声明上方。

14.2　Struts2 数据校验与拦截器

实例 356	日期转换器 光盘位置：光盘\MR\14\356	中级 实用指数：★★★

实例说明

本实例实现的是日期转换器。当用户在注册页面中输入日期（如 1989-09-02）时，单击“注册”按钮，将

会跳转到注册成功页面，同时会将输入的日期格式进行转换并显示在页面上，如图 14.5 所示。

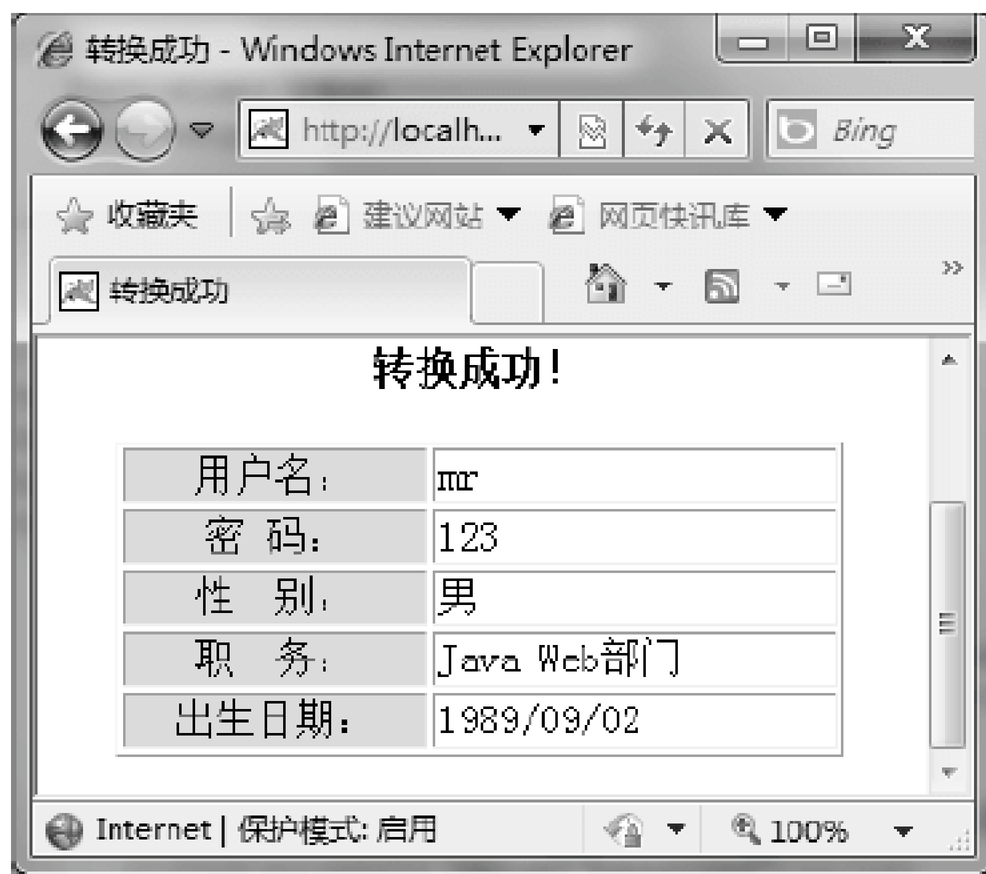


图 14.5　日期的转换

关键技术

要实现日期格式的转换，就要自定义一个日期转换器。在该日期转换器中使用 SimpleDateFormat 类进行日期格式的转换。SimpleDateFormat 类中常用的方法如下。

- public SimpleDateFormat(String pattern)：该构造方法可以用参数 pattern 指定格式。pattern 是由普通字符和一些称为格式符的符号组成的字符序列。
- public String format(Date date)：用构造方法创建的对象，同时用该对象调用此方法来格式化时间对象 date。

设计过程

（1）在项目中创建日期转换器类 UserConverter，在该类中实现对日期的转换。具体代码如下：

```
public class UserConverter extends StrutsTypeConverter
{
	public Object convertFromString(Map context, String[] values,			//实现将字符串类型转换成复合类型的方法
			Class toClass){
			User user = new User();						//创建一个 User 实例
			SimpleDateFormat format1=new SimpleDateFormat("yyyy-MM-dd");
			try {
				Date date1 = format1.parse(values[0]);
				SimpleDateFormat format2=new SimpleDateFormat("yyyy/MM/dd");
				String birthday=format2.format(date1);
				user.setDate(birthday);					//为 User 实例赋值
				} catch (ParseException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			return user;							//返回转换来的 User 实例
	}
	public String convertToString(Map context, Object o){			//实现将复合类型转换成字符串类型的方法
			User user = (User) o;						//将需要转换的值强制类型转换为 User 实例
			return "<" + user.getDate()+">";
	}
}
```

（2）把日期转换器类配置到 xwork-conversion.properties 中，具体代码如下：

```
lee.User=lee.UserConverter
```

秘笈心法

心法领悟 356：xwork-conversion.properties 文件中类的写法。

在该文件中，name 值是指待转换的类，该类一定要写成“包名.类名”的形式；Value 值则是自定义的类型转换器，同样也要写成“包名.类名”的形式。

实例 357	实现空表单信息的提示 光盘位置：光盘\MR\14\357	高级 实用指数：★★★★

实例说明

本实例实现的是空表单信息的提示。在文本框中不输入任何内容，单击“提交”按钮，会显示校验信息，

如图 14.6 所示。

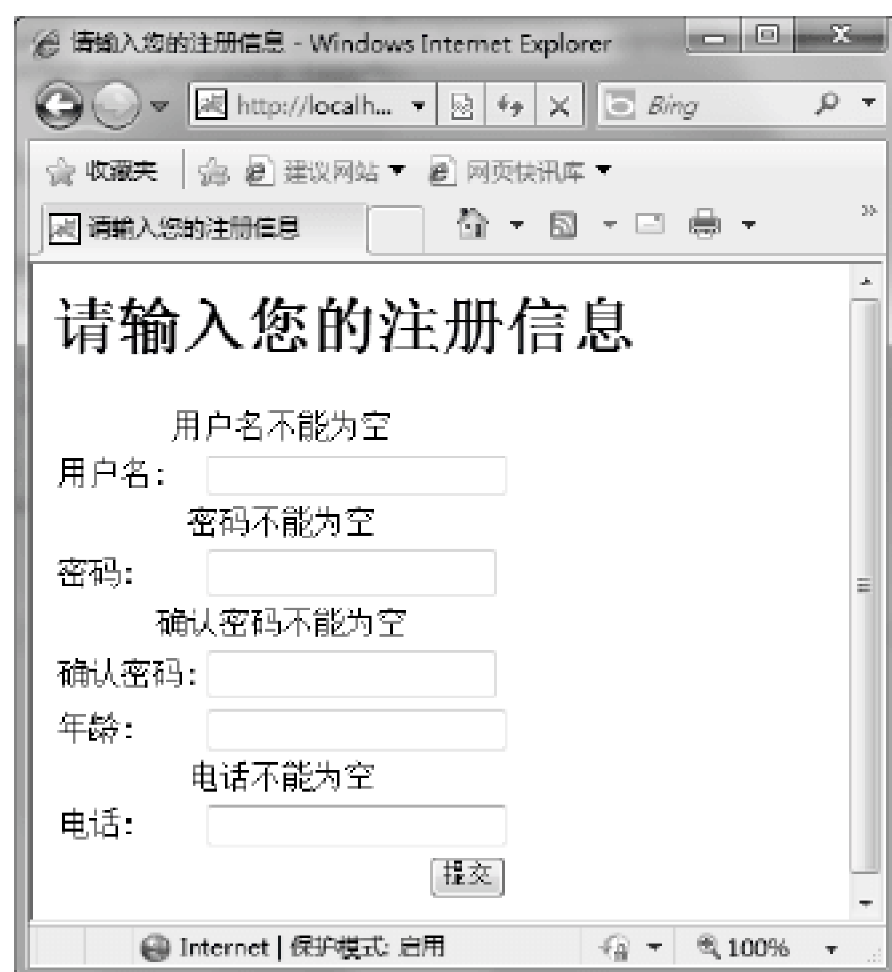


图 14.6　空表单信息提示

关键技术

在 Struts2 框架中提供了服务器端数据验证机制，其验证方式非常灵活，可分为手动验证与使用验证框架验证，本实例采用的是验证框架。

Struts2 提供了多个校验器，下面列举一些常用的校验器。

- required：必填校验器。
- requiredstring：必填字符串校验器。
- int：整数校验器。
- double：双精度浮点数校验器。
- date：日期校验器。
- expression：表达式校验器。
- email：电子邮件校验器。
- regex：正则表达式校验器。

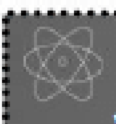
设计过程

在项目中创建 ZhuCeAction 类，然后为该 Action 指定一个校验文件 ZhuceAction-validation.xml。该校验文件的具体代码如下：

```
<validators>
	<field name="name">                                    //校验用户名不能为空
	    <field-validator type="requiredstring">
	            <param name="trim">true</param>
	            <message>用户名不能为空</message>
	        </field-validator>
	</field>
	<field name="pass">                                    //校验密码不能为空
	    <field-validator type="requiredstring">
	            <param name="trim">true</param>
	            <message>密码不能为空</message>
	        </field-validator>
	</field>
	        <field name="rpass">                           //校验确认密码不能为空
	    <field-validator type="requiredstring">
	            <param name="trim">true</param>
	            <message>确认密码不能为空</message>
	        </field-validator>
	        <field-validator type="fieldexpression">       //校验密码输入的一致性
	            <param name="expression"><![CDATA[(pass==rpass)]]></param>
	            <message>两次输入密码不一致</message>
	        </field-validator>
	</field>
	        <field name="phone">                           //校验电话不能为空
	                <field-validator type="requiredstring">
	            <param name="trim">true</param>
	            <message>电话不能为空</message>
	        </field-validator>
	        <field-validator type="regex">                 //校验输入电话的格式
	            <param name="expression"><![CDATA[(1\d{10})]]></param>
	            <message>电话必为 11 位数字，且必以 1 开头</message>
	        </field-validator>
	</field>
</validators>
```

心法领悟 357：校验文件的命名规范。

校验文件存放的路径要和校验的 Action 路径一致，要命名为 Action 类名-validation.xm 的形式。



实例 358　计时拦截器

光盘位置：光盘\MR\14\358

高级

实用指数：★★★

实例说明

本实例实现的是计时拦截器。用户在登录页面中填写用户名，然后单击“提交”按钮，在控制台中会显示程序运行的时间，如图 14.7 所示。

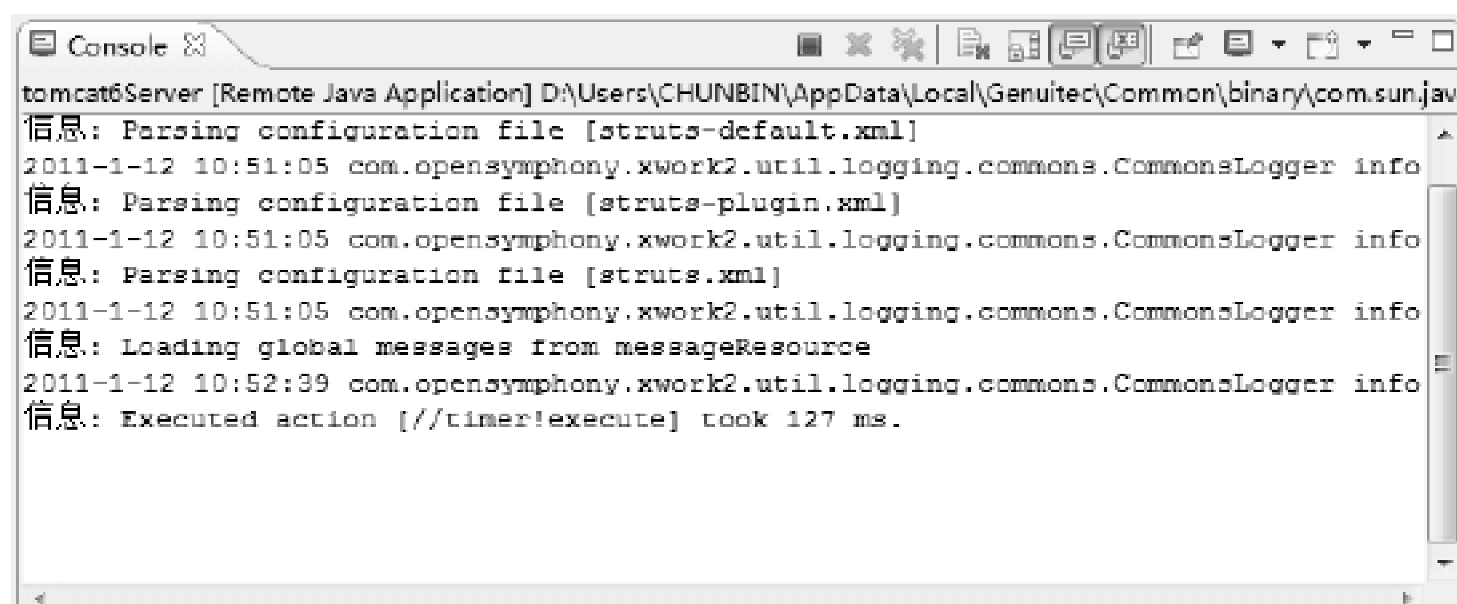


图 14.7　计时拦截器

关键技术

要实现计时拦截器的功能，需要在 struts.xml 中配置相应的内置拦截器 timer。timer 拦截器能够统计每个 Action 方法运行所需的时间，并将时间差打印出来。

Struts2 常用的内置拦截器介绍如下。

- chain：将所有属性从前一个 Action 复制到当前 Action 中，一般是和 struts.xml 中配置的 type="chain" 一起使用。
- exeception：提供了异常处理的核心功能，允许把一个异常映射到一个结果码上，就像是 Action 中返回的结果码，而不是抛出异常。
- i18n：用于支持国际化，将当前会话选择的 locale 放入用户的 session 中。
- validation：调用验证框架读取 xxAction-validation.xml 文件并执行其中的内容。
- token：检查 token 值的有效性，用于防止表单的重复提交。
- logger：记录一个 Action 中执行的开始与结束。
- fileUpLoad：用于支持文件上传。
- timer：输出 Action 执行的时间。

设计过程

在 struts.xml 的<action>标签中配置 Struts2 提供的内置拦截器 timer，具体代码如下：

```
<struts>
    <package name="lee" extends="struts-default">
            <action name="timer" class="lee.TimerAction">            //定义名为 timer 的 Action，其实现类为 lee.TimerAction
             <interceptor-ref name="timer"/>                         //使用计时的 timer 拦截器
                 <result name="success">/welcome.jsp</result>
            </action>
    </package>
</struts>
```

秘笈心法

心法领悟 358：计时拦截器的作用。

计时拦截器是显示执行 Action 所需的时间，在评估一个系统的性能方面很有用。



实例 359　等待拦截器

光盘位置：光盘\MR\14\359

高级

实用指数：★★★

实例说明

本实例实现的是等待拦截器。在地址栏中输入 http://localhost:8080/359/waiting.action，则会显示一个正在加

载的页面。等待 5 秒后加载完成，同时会显示翼飞图书管理系统的主页，如图 14.8 所示。



图 14.8　等待拦截器

关键技术

要实现等待拦截器的功能，需要在 struts.xml 中配置相应的内置拦截器 execAndWait。若服务器比较繁忙，浏览器访问时会长时间地显示空白页面。execAndwait 可以解决这个难题，它能够在此过程中显示一个服务器繁忙的页面。

设计过程

在 struts.xml 的 action 标签中配置 Struts2 提供的内置拦截器 execAndWait，具体代码如下：

```
<struts>
  <package name="default" extends="struts-default">
    <action name="waiting" class="cn.mrcast.action.WaitAction">          //定义名为 waiting 的 Action，其实现类为 cn.mrcast.action.TimerAction
      <interceptor-ref name="execAndWait"/>                              //使用等待的 execAndWait 拦截器
      <result name="wait">/waiting.jsp</result>                          //服务器繁忙页面
      <result name="success">/success.jsp</result>                       //加载成功页面
    </action>
  </package>
</struts>
```

秘笈心法

心法领悟 359：显示繁忙页面的其他方式。

本实例只是 Struts2 内置的显示繁忙页面的一种简单方式，在实际的开发过程中也可以使用 Ajax 技术来实现，这种技术可以做到页面的无刷新操作，同时还可以通过进度条实时显示页面加载的进度。

实例 360	权限验证拦截器 光盘位置：光盘\MR\14\360	中级 实用指数：★★★

实例说明

本实例主要实现的是如果用户没有登录就无法访问网站的其他内容，如果强行进入则会跳转到登录页面并

且给出提示，如图 14.9 所示。

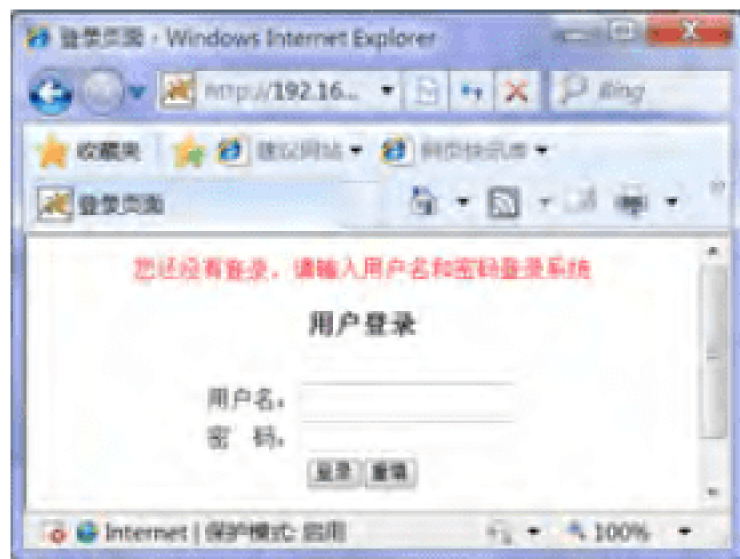


图 14.9　权限验证拦截器

关键技术

对于权限拦截器，都是通过跟踪用户的 Session 来完成的，通过 ActionContext 可以访问到 Session 中的属性，拦截器中方法的 invocation 参数同样也就可以访问请求中的 ActionContext。本实例中拦截器的主要代码如下：

```
public class AuthorityInterceptor extends AbstractInterceptor
{
      public String intercept(ActionInvocation invocation) throws Exception
         {
                  ActionContext ctx = invocation.getInvocationContext();
                  Map session = ctx.getSession();
                  String user = (String)session.get("user");
                  if (user != null && user.equals("mr") )
                  {
                          return invocation.invoke();
                  }
                  ctx.put("zx" , "您还没有登录，请输入用户名和密码登录系统");
                  return Action.LOGIN;
      }
}
```

设计过程

（1）创建拦截器类，在其中编写相关代码实现对权限的验证，以及对请求结果的判定。具体代码如下：

```
public class AuthorityInterceptor extends AbstractInterceptor
{
      public String intercept(ActionInvocation invocation) throws Exception
         {
                  ActionContext ctx = invocation.getInvocationContext();
                  Map session = ctx.getSession();
                  String user = (String)session.get("user");
                  if (user != null && user.equals("mr") )
                  {
                          return invocation.invoke();
                  }
                  ctx.put("zx" , "您还没有登录，请输入用户名和密码登录系统");
                  return Action.LOGIN;
      }
}
```

（2）在 struts.xml 中配置拦截器，具体代码如下：

```
<action name="viewBook">
                  <result>/viewBook.jsp</result>
                          <!-- 拦截器一般配置在 result 元素之后! -->
                          <interceptor-ref name="defaultStack"/>
                          <interceptor-ref name="authority"/>
            </action>
```

心法领悟 360：权限的验证。

对于用户的权限验证，不仅是使用简单的权限拦截器，有时为了安全性更好，还要使用 JS 代码进行一些验证以及与数据库相连实现一些验证等。

14.3　文件上传与下载

实例 361　单文件的上传

光盘位置：光盘\MR\14\361

高级

实用指数：★★★

实例说明

本实例实现的是单文件上传。在页面中输入个人信息并选择要上传的信息后，单击“提交”按钮，即可将文件成功上传，如图 14.10 所示。

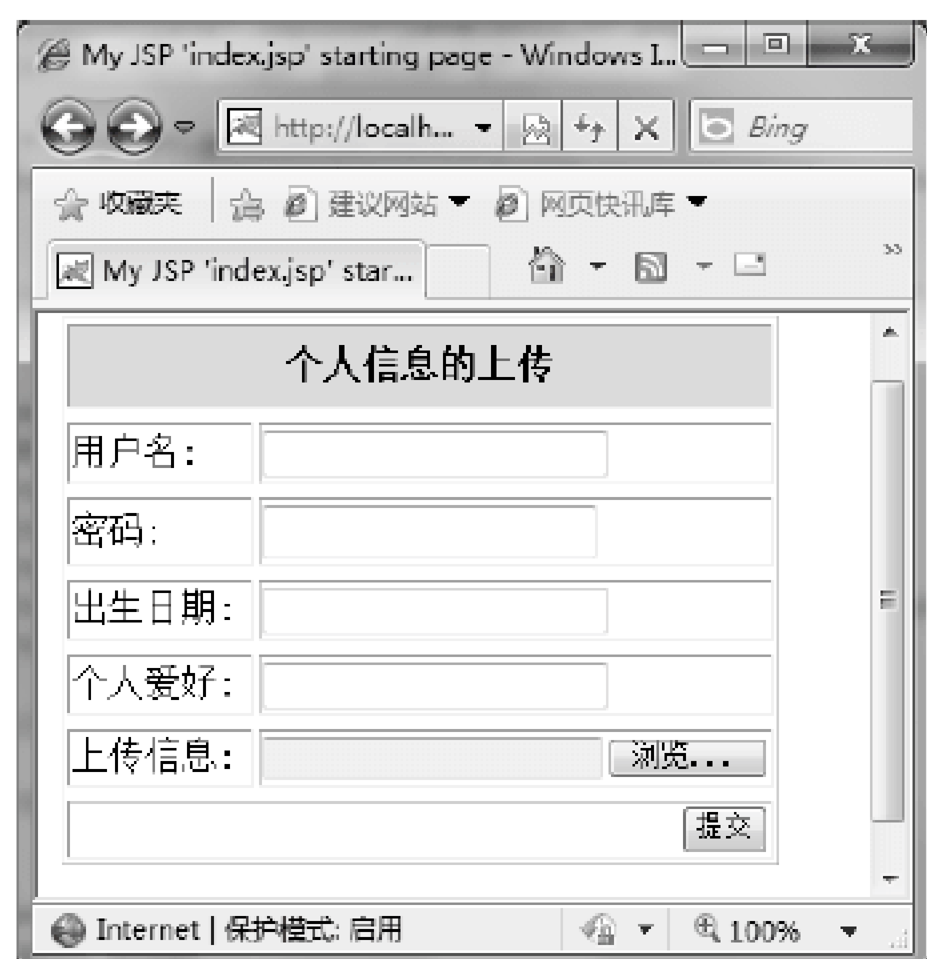


图 14.10　单文件的上传

关键技术

如果表单中包含一个 name 属性为 xx 的文件域，Struts2 将提供如下 3 个属性对上传的文件域进行封装。

- 类型为 File 的 xx 属性封装该文件域对应的文件内容。
- 类型为 String 的 xxFileName 属性封装该文件域对应的文件名。
- 类型为 String 的 xxContentType 属性封装该文件域对应的文件类型。

通过以上 3 个属性可以直接使用 getxx()方法获取上传文件的文件内容、文件名和文件类型。

在项目中创建 UpLoadAction 类，在该类中对上传的文件进行相应的处理。具体代码如下：

```
private File pic;                                          //封装文件内容
 private String picFileName;                               //封装文件名
 private String picContentType;                            //封装文件类型
 public File getPic() {
        return pic;
 }
 public void setPic(File pic) {
        this.pic = pic;
 }
 public String getPicFileName() {
        return picFileName;
 }
 public void setPicFileName(String picFileName) {
        this.picFileName = picFileName;
 }
 public String getPicContentType() {
        return picContentType;
 }
 public void setPicContentType(String picContentType) {
        this.picContentType = picContentType;
 }
 public String exec()
 {
        return "input";
 }
 public String Up()
 {
        File sa=new File(ServletActionContext.getServletContext().getRealPath("up"),picFileName);
        InputStream in=null;                               //定义输入流
        OutputStream ou=null;                              //定义输出流
        try {
               sa.getParentFile().mkdirs();
               in=new FileInputStream(pic);
```

```
            ou=new FileOutputStream(sa);
            byte[] b=new byte[1024];
            int len=0;
          while((len=in.read(b))!=-1)
            {
              ou.write(b,0,len);
            }
            in.close();
            ou.close();
    } catch (IOException e) {
            e.printStackTrace();
    }
    return "success";
}
```

秘笈心法

心法领悟 361：上传文件表单属性的使用。

在用 Struts2 上传文件时，form 表单中除了要设置 action 属性，还要设置 enctype 的属性为 multipart/form- data。enctype 属性的默认值是 application/x-www-form-urlencoded。这种编码使用的是有限的字符集，当使用了数字字符或者非字母时，必须用%HH 代替（其中的 H 指十六进制数字）。



实例 362　**上传错误信息的提示**　高级

光盘位置：光盘\MR\14\362　实用指数：★★★

实例说明

本实例实现的是上传错误信息的提示。在页面中输入个人信息并选择要上传的信息，然后单击“提交”按钮，当上传文件的大小超过 Struts2 中默认设置的文件大小时，就会显示上传错误的信息，如图 14.11 所示。

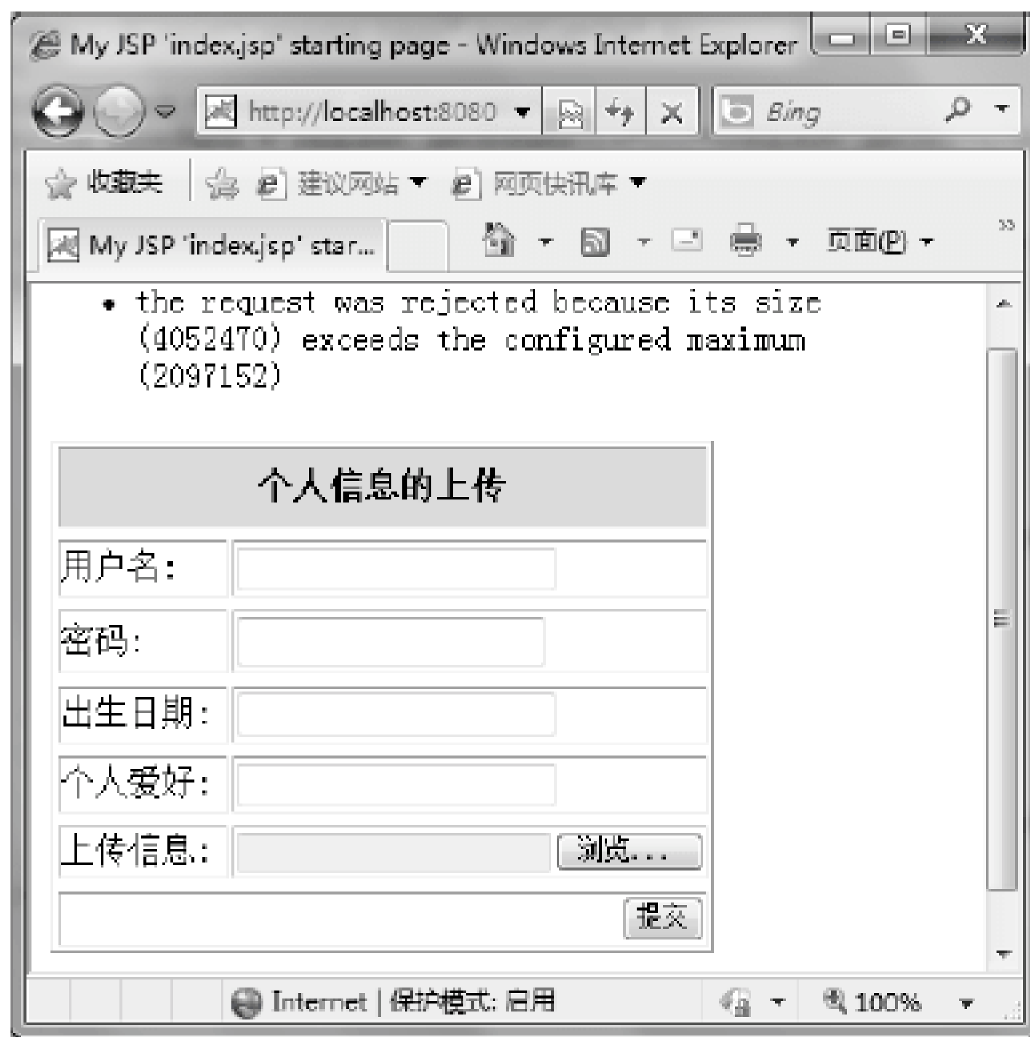


图 14.11　上传错误信息的提示

关键技术

要实现上传错误信息的提示，可以使用 Struts2 标签库中的标签进行显示。对于信息的显示，Struts2 提供了以下 3 个标签。

- ❑ fielderror：显示数据校验错误信息。
- ❑ actionerror：显示 Action 中的错误信息。
- ❑ actionmessage：显示 Action 中的信息。

对于本实例，可以采用 actionerror 标签显示 Action 中的错误信息。

在项目中创建 UpLoadAction 类，在该类中对上传的文件进行相应的处理，然后创建用于上传文件的 index.jsp 页面。具体代码如下：

```
<s:actionerror/>                                                          //显示错误信息
<body>
    <s:form action="upic" enctype="multipart/form-data" method="post">     //设置提交方式
    <div align="center">
       <table width="317" border="1">
           <tr bgcolor="#FFCCFF">
             <td height="38" colspan="2"> <div align="center"><strong>个人信息的上传  </strong></div></td>
           </tr>
           <tr>
             <td width="83"><s:textfield name="username" label="用户名"/></td>
```

```
            </tr>
            <tr>
              <td> <s:password name="password" label="密码" /></td>
            </tr>
            <tr>
              <td><s:textfield name="date" label="出生日期"/></td>
            </tr>
            <tr>
              <td><s:textfield name="like" label="个人爱好"/></td>
            </tr>
            <tr>
              <td> <s:file name="pic" label="上传信息" /></td>
            </tr>
            <tr>
              <td><s:submit value="提交" method="Up"/></td>
            </tr>
          </table>
        </div>
        <div align="center"></div>
        </s:form>
</body>
```

秘笈心法

心法领悟 362：上传错误信息的显示方式。

如果在页面上没有设置 actionerror 标签，则上传的错误信息会直接输出到控制台上。

实例 363　特定文件格式的上传

光盘位置：光盘\MR\14\363

高级

实用指数：★★★

实例说明

本实例实现的是特定文件格式的上传。在本实例中限定了只能上传图片格式的文件，单击“浏览”按钮，当上传的不是图片格式的文件时，会显示上传错误的信息，如图 14.12 所示。



图 14.12　特定文件格式的上传

关键技术

要实现特定文件格式的上传，就要用到 Struts2 自带的 fileUpLoad 拦截器。该拦截器用于拦截文件上传的请求，通过设置拦截器的参数，可以限制上传文件的类型、大小。

对于 fileUpLoad 拦截器，可以为其指定以下两个参数。

- allowTypes：用于指定允许上传的文件类型。
- maximumSize：用于指定允许上传的文件大小。

在项目中创建 UpLoadAction 类，在该类中对上传的文件进行相应的处理，然后创建 struts.xml 文件，用于对上传文件的格式进行限定。具体代码如下：

```
<struts>
  <package name="default" extends="struts-default">
    <action name="upic" class="cn.mrcast.action.UpLoadAction">
        <interceptor-ref name="fileUpload">
            <param name="allowedTypes">image/bmp,image/x-png,image/gif,image/jpeg</param>          //对上传文件的格式进行限定
        </interceptor-ref>
        <interceptor-ref name="defaultStack"></interceptor-ref>
            <result name="success">/success.jsp</result>
            <result name="input">/index.jsp</result>
```

```
        </action>
    </package>
</struts>
```

秘笈心法

心法领悟 363：通过 allowedTypes 属性设置多种文件类型。

当设置多种文件类型时，各类型之间用“,”隔开。

实例 364　限定上传文件的大小

光盘位置：光盘\MR\14\364

高级

实用指数：★★★

实例说明

本实例实现的是限定上传文件的大小。在本实例中限定了只能上传大小在 10090 字节以下的文件，单击“浏览”按钮，当上传文件的大小大于 10090 字节时，会显示上传错误的信息，如图 14.13 所示。



图 14.13　限定上传文件的大小

关键技术

要实现限定上传文件的大小，就要用到 Struts2 自带的 fileUpLoad 拦截器。该拦截器用于拦截文件上传的请求，通过设置拦截器的参数，可以限制上传文件的类型、大小。

设计过程

在项目中创建 UpLoadAction 类，在该类中对上传的文件进行相应的处理；然后创建 struts.xml 文件，用于对上传文件的大小进行限定。具体代码如下：

```
<struts>
    <package name="default" extends="struts-default">
        <action name="upic" class="cn.mrcast.action.UpLoadAction">
          <interceptor-ref name="fileUpload">
             <param name="maximumSize">10090</param>                //对上传文件的大小进行限定
         </interceptor-ref>
         <interceptor-ref name="defaultStack"></interceptor-ref>
         <result name="success">/success.jsp</result>
         <result name="input">/index.jsp</result>
        </action>
    </package>
</struts>
```

秘笈心法

心法领悟 364：上传文件大小的单位。

上传文件大小的单位是字节。

实例 365　多文件的上传

光盘位置：光盘\MR\14\365

高级

实用指数：★★★

本实例实现的是多文件的上传。运行程序，在如图 14.14 所示页面中单击每个“浏览”按钮，选择要上传的文件，然后单击“确定”按钮，即可将多个文件成功上传。

关键技术

要实现多文件的上传，就要使用数组。Struts2 负责将 3 个文件域对应的文件内容、文件名、文件类型分别放入 Action 的文件内容数组、文件名数组和文件类型数组中。

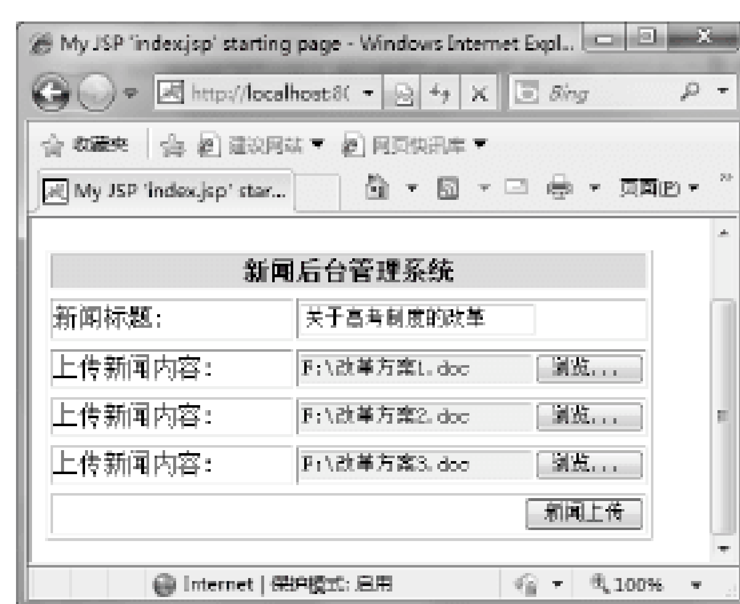


图 14.14　多文件上传页面

设计过程

在项目中创建类 UpLoadAction，在该类中对上传的文件进行相应的处理。具体代码如下：

```
private File[] pic;                                    //使用数组封装多个文件内容
    private String[] picFileName;                      //使用数组封装多个文件名
    private String[] picContentType;                   //使用数组封装多个文件类型
    public File[] getPic() {
        return pic;
    }
    public void setPic(File[] pic) {
        this.pic = pic;
    }
    public String[] getPicFileName() {
        return picFileName;
    }
    public void setPicFileName(String[] picFileName) {
        this.picFileName = picFileName;
    }
    public String[] getPicContentType() {
        return picContentType;
    }
    public void setPicContentType(String[] picContentType) {
        this.picContentType = picContentType;
    }
    public String Up()
    {

        for(int i=0;i<pic.length;i++)
        {
        File sa=new File(ServletActionContext.getServletContext().getRealPath("up"),picFileName[i]);
        InputStream in=null;                           //定义输入流
        OutputStream ou=null;                          //定义输出流
        try {
            sa.getParentFile().mkdirs();
            in=new FileInputStream(pic[i]);
            ou=new FileOutputStream(sa);
            byte[] b=new byte[1024];
            int len=0;
           while((len=in.read(b))!=-1)
             {
              ou.write(b,0,len);
             }
            in.close();
            ou.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
     }
        return "success";
    }
```

秘笈心法

心法领悟 365：对于 file 标签 name 属性的设置。

为了让数组一次性地封装 3 个文件域，需要将上面 3 个文件域的 name 设置为相同的名称。

实例 366　文件下载　高级
光盘位置：光盘\MR\14\366　实用指数：★★★

实例说明

本实例实现的是文件下载。运行程序，在如图 14.15 所示的页面中单击“衣服图片”右侧的“下载”按钮，即可将需要下载的图片显示到页面上。



图 14.15　文件下载页面

关键技术

要实现文件的下载，需要在 struts.xml 中配置一个类型为 stream 的结果。它有 4 个属性，分别介绍如下。

- contentType：指定被下载文件的类型。
- inputName：指定被下载文件的入口输入流。
- contentDisposition：指定下载的文件名。
- bufferSize：指定下载文件时的缓冲大小。

（1）在项目中创建类 LoadAction，在该类中主要实现返回一个 InputStream 流的方法。具体代码如下：

```
private String filename;                                          //定义文件名
  private String Path;                                            //定义文件路径
  private String contentType;                                     //定义文件类型
  private InputStream inputStream;                                //定义流
      public void setInputStream(InputStream inputStream) {
        this.inputStream = inputStream;
  }
  public String getFilename() {
        return filename;
  }
  public void setFilename(String filename) {
        this.filename = filename;
         }
  public String getPath() {
        return Path;
  }
  public void setPath(String path) {
        Path = path;
  }
  public String getContentType() {
        return contentType;
  }
  public void setContentType(String contentType) {
        this.contentType = contentType;
  }
  public InputStream getInputStream() throws Exception
  {
        return ServletActionContext.getServletContext().getResourceAsStream(Path);     //返回一个 InputStream 流
  }
  public String execute() throws Exception {
        Path="/image/choice.jpg";                                 //要下载的文件名
        filename="test.jpg";                                      //保存文件时的名称
        contentType="/gif";                                       //保存文件的类型
        return SUCCESS;
  }
```

（2）在 struts.xml 中配置一个类型为 stream 的结果。具体代码如下：

```
<struts>
 <package name="default" extends="struts-default">
  <action name="lo" class="cn.mrcast.action.LoadAction">
    <result name="success" type="stream">                         //配置类型 stream
    <param name="contentType">${contentType}</param>              //配置文件类型
    <param name="inputName">inputStream</param>                   //配置输入流
    <param name="contentDisposition">filename="${filename}"</param>   //配置文件名
    <param name="bufferSize">2048</param>                         //配置缓冲大小
    </result>
  </action>
 </package>
</struts>
```

秘笈心法

心法领悟 366：struts.xml 中的配置。

在对类型为 stream 的结果进行配置时，将直接向客户端返回一个输入流，因此也就无须指定 location 属性。

14.4　Struts2 对 Ajax 的支持

实例 367	调试信息的输出 光盘位置：光盘\MR\14\367	中级 实用指数：★★★

实例说明

本实例实现的是调试信息的输出。运行程序，单击页面中的 Debug 超链接，将会显示大量的信息，如图 14.16 所示。

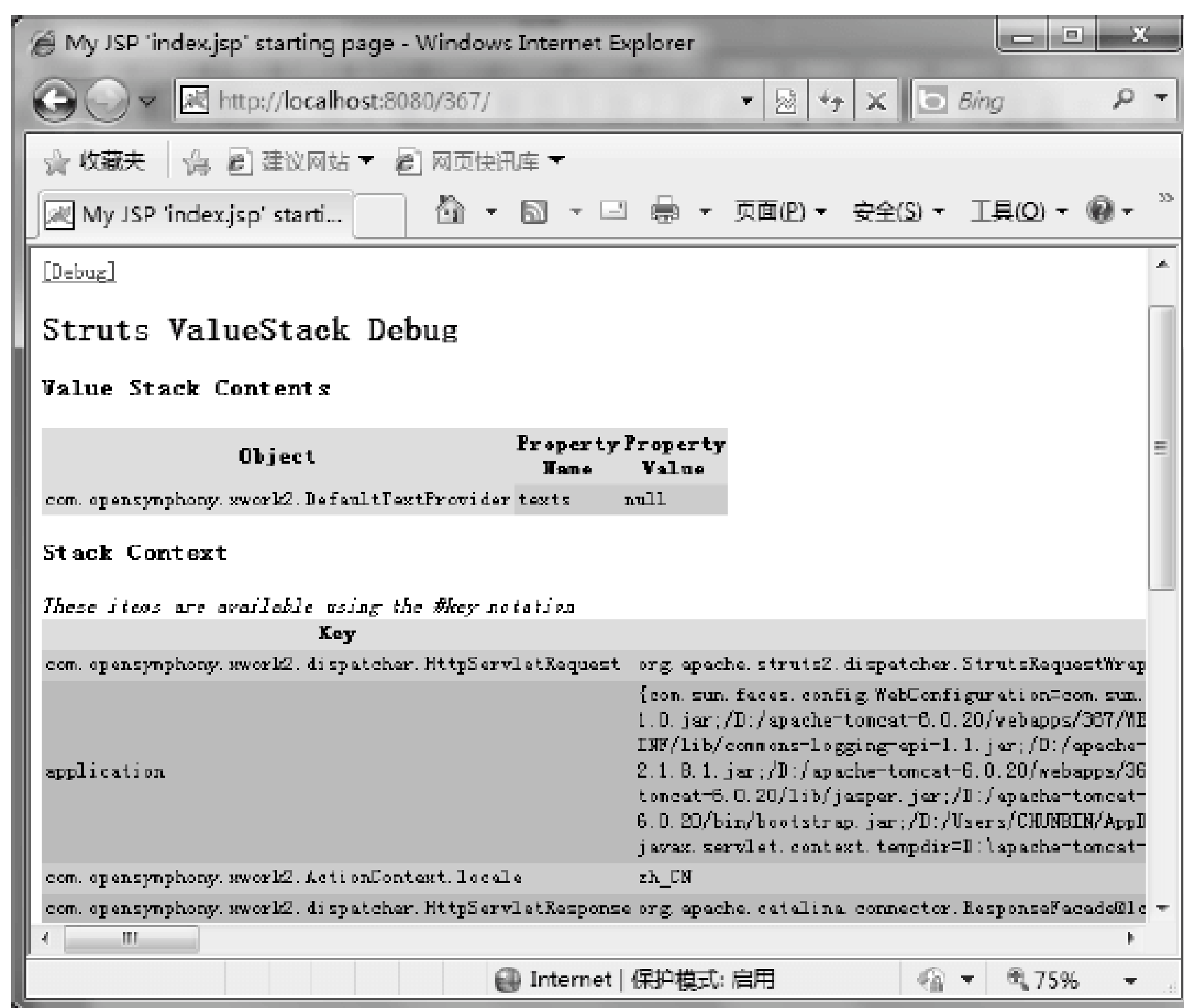


图 14.16　调试信息的输出

关键技术

要实现调试信息的输出，就要用到 Struts2 自带的标签<s:debug />。该标签用于显示服务器、Action 的有关信息，一般是为了调试程序使用。

创建用于输出 debug 信息的 index.jsp 页面。具体代码如下：

```
<body>
    调试信息的输出：
    <s:debug></s:debug>                    //输出调试信息
</body>
```

秘笈心法

心法领悟 367：debug 标签中的属性。

debug 标签中只有一个 id 属性，该属性本身是没有什么意义的，只是该元素的一个引用 id。

实例 368　数据校验错误信息的输出

光盘位置：光盘\MR\14\368

中级

实用指数：★★☆

实例说明

本实例实现的是数据校验错误信息的输出。运行程序，当不输入任何用户名和密码时，单击页面中的“提交”按钮，将会显示相应的数据检验错误信息，如图 14.17 所示。

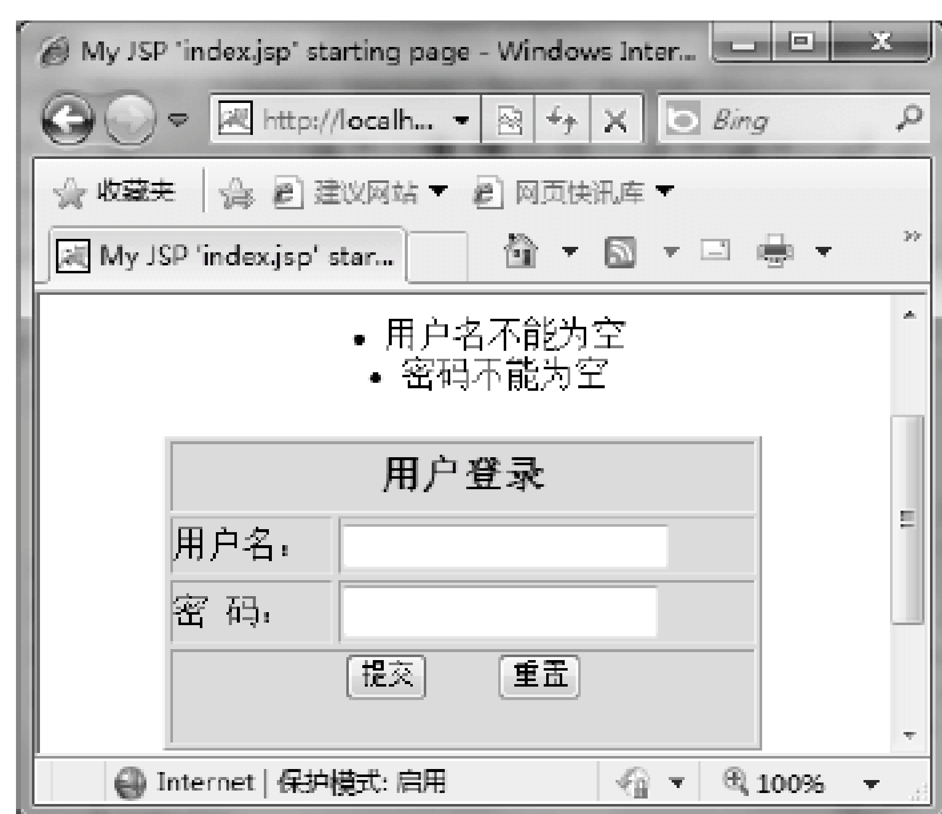


图 14.17　数据校验错误信息的输出

关键技术

要实现数据校验错误信息的输出，就要用到 Struts2 自带的标签 <s:fielderror/>。fielderror 标签输出 Action 的 fieldErrors 属性保存的字段错误信息，fieldErrors 是一个 Map 类型的属性。

在项目中创建类 ValidateAction，该类主要用于添加 fielderror 信息。具体代码如下：

```
private String username;
 private String password;
 public void setUsername(String username) {
        this.username = username;
 }
 public void setPassword(String password) {
        this.password = password;
 }
 public String getUsername() {
        return username;
 }
 public String getPassword() {
        return password;
 }
 public String Validate()
        {
    if(username.equals("")||password.equals("")){
          this.addFieldError("username", "用户名不能为空");              //添加用户名不能为空信息
          this.addFieldError("password", "密码不能为空");                //添加密码不能为空信息
          return "input";                                                //返回到登录页面
    }
    else {
            return "success";                                            //返回到成功页面
    }
    }
```

秘笈心法

心法领悟 368：特定字段错误信息的输出。

对于 fielderror 标签，可以在其中嵌套 param 标签来指定输出特定字段的错误信息。



实例 369 Action 中错误信息的输出

光盘位置：光盘\MR\14\369

中级

实用指数：★★☆

实例说明

本实例实现的是 Action 中错误信息的输出。运行程序，当输入的用户名不为 mr 或者密码不为 mrsoft 时，单击页面上的“提交”按钮，将会显示相应的 Action 错误信息，如图 14.18 所示。



图 14.18 Action 中错误信息的输出

关键技术

要实现 Action 中错误信息的输出，就要用到 Struts2 自带的标签<s:actionerror/>。actionerror 标签输出 Action 的 actionErrors 属性保存的错误信息，actionErrors 是一个 Collection 类型的属性。

设计过程

在项目中创建类 ErrorAction，该类主要用于添加 actionerror 信息。具体代码如下：

```
private String username;
    private String password;
      public void setUsername(String username) {
          this.username = username;
    }
    public void setPassword(String password) {
          this.password = password;
    }
    public String getUsername() {
          return username;
    }
    public String getPassword() {
          return password;
    }
    public String Error(){
    if(!username.equals("mr")||!password.equals("mrsoft")){                //判断用户名和密码的正确性
      this.addActionError("用户名或密码错误");                              //添加错误信息
      return "input";                                                       //返回到登录页面
        }
      else{
            return "success";                                               //返回到成功页面
        }
}
```

秘笈心法

心法领悟 369：actionerror 标签和 actionmessage 标签的区别。

actionerror 标签用于输出 Action 的错误消息，而 actionmessage 标签则用于输出 Action 的一般性消息。



实例 370 显示 Action 的信息

光盘位置：光盘\MR\14\370

中级

实用指数：★★☆

实例说明

本实例实现的是显示 Action 的信息。运行程序，单击页面中的“提交”按钮，将会显示相应的 Action 信息，如图 14.19 所示。



图 14.19　显示 Action 信息

关键技术

要实现显示 Action 的信息，就要用到 Struts2 自带的标签<s:actionmessage/>。actionmessage 标签输出 action 的 actionMessages 属性保存的消息，actionMessages 是一个 Collection 类型的属性。

设计过程

在项目中创建类 MessageAction，该类主要用于添加 actionmessage 信息。具体代码如下：

```
private String username;
      private String password;
      public String getUsername() {
          return username;
      }
      public void setUsername(String username) {
          this.username = username;
      }
      public String getPassword() {
          return password;
      }
      public void setPassword(String password) {
          this.password = password;
      }
      public String Message()
    {
          if(!username.equals("")||!password.equals("")){                    //判断是否输入用户名或密码
          this.addActionMessage("你输入的用户名是:   "+username);              //添加用户名信息
          this.addActionMessage("你输入的密码是:   "+password);                //添加密码信息
              }
          return "input";
    }
```

秘笈心法

心法领悟 370：输出消息的布局。

fielderror、actionerror、actionmessage 标签输出消息的布局依赖于当前各自页面中使用的主题。

实例 371　显示新闻列表

光盘位置：光盘\MR\14\371

中级

实用指数：★★☆

本实例实现的是显示新闻列表。运行程序，可以看到一个新闻管理平台的页面，过了 2 秒后页面会无刷新地显示新闻列表，如图 14.20 所示。



图 14.20　显示新闻列表

关键技术

实现无刷新显示新闻列表，要使用 div 标签生成 div 元素。该 div 元素中并不是静态内容，而是从服务器中获取到的数据。为了获取服务器中的数据，div 标签中提供了 href 属性来生成 div 元素的内容。

div 标签中提供了两个属性用于指定更新延迟和更新频率，分别介绍如下。

- delay：更新 div 内容的时间延迟，单位是毫秒。
- updateFreq：更新 div 内容的时间间隔，单位是毫秒。

设计过程

创建用于请求 Action 的 index.jsp 页面，同时设置更新 div 内容的时间延迟为 2 秒。具体代码如下：

```
<s:url id="te" value="/new.action"/>                                        //用于请求 Action
 <span class="STYLE1">
     <sx:div href="%{te}" delay="2000">                                      //设置 div 的 href 属性
       <div align="center">
         <p> </p>
         <p><strong>欢迎来到新闻管理平台，新闻列表即将显示！</strong></p>
       </div>
     </sx:div>
  </span>
```

秘笈心法

心法领悟 371：div 标签的 href 属性。

该属性必须是一个 Action，由该 Action 负责生成 div 元素的内容。

实例 372 页面的自动刷新

光盘位置：光盘\MR\14\372

中级

实用指数：★★★

实例说明

本实例实现的是页面的自动刷新。运行程序，可以看到一个初始页面。停留一段时间以后，该页面转向一个计数器页面，并且是每隔 4 秒进行一次刷新，如图 14.21 所示。

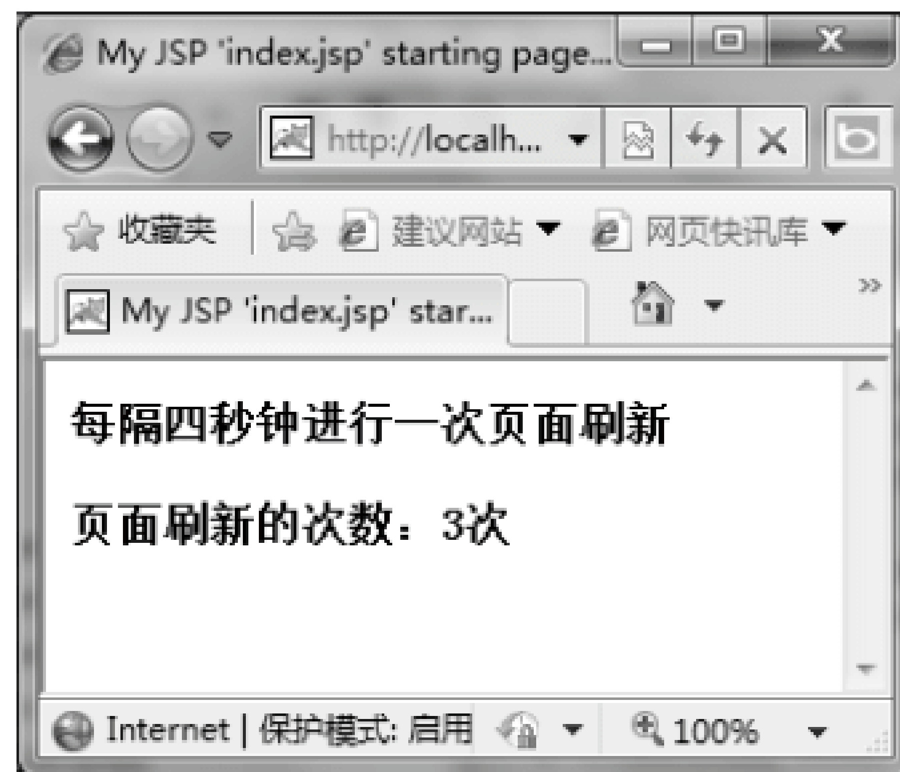


图 14.21　页面自动刷新

关键技术

实现页面的自动刷新，要用到 div 标签中的 updateFreq 属性，该属性用于设置更新 div 内容的时间间隔。

创建用于请求 Action 的 index.jsp 页面，同时设置更新 div 内容的时间间隔为 4 秒。具体代码如下：

```
<body>
  <s:url id="te" value="/cou.action"/>                                              //请求一个 Action
    <p>
      <sx:div href="%{te}" updateFreq="4000" indicator="first"></sx:div>            //设置更新 div 内容的时间间隔为 4 秒
    </p>
    <br/>
    <p class="STYLE1" id="first">这是初始页面</p>
</body>
```

秘笈心法

心法领悟 372：updateFreq 属性的使用。

在 div 标签中如果不指定该属性，则只在页面加载时更新该 div 的内容。

实例 373　访问注册页面出错

光盘位置：光盘\MR\14\373

中级

实用指数：★★☆

实例说明

本实例实现的是访问注册页面出错提示。运行实例，即可进入用户登录页面。稍等 1 秒后，将会自动跳转到注册页面。当注册页面不存在时，程序会提示出错信息，如图 14.22 所示。

图 14.22　访问注册页面出错提示

关键技术

要实现访问注册页面出错提示，需要在<sx:div>标签中使用 errorText 属性，该属性用于设置出错信息的内容，为了在本实例中看到效果，同时在代码中设置了 delay 属性，该属性用于指定更新 div 内容的时间延迟，单位是毫秒。

设计过程

创建用户登录页面 index.jsp，在该页面中主要显示用户登录信息以及负责跳转到注册页面。具体代码如下：

```
<body bgcolor="#FFFFFF" leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
  <p>
    <s:url id="test" value="/Test.jsp"></s:url>                                    //跳转页面的 URL
    <sx:div id="error" href="%{test}" delay="1000" errorText="访问注册页面出错，你所请求的资源不存在"></sx:div>    //提示错误信息
</p>
  <table width="380" border="1" bgcolor="#FFCCFF">                                    //设置登录信息
  <tr>
    <td height="28" colspan="2" bgcolor="#FF0000"><div align="center"><strong>页面即将跳转到用户注册页面，请耐心等待...</strong></div></td>
  </tr>
  <tr>
    <td height="24" colspan="2"><div align="center" class="STYLE1">用户登录</div></td>
    </tr>
  <tr>
    <td width="143" height="17"><div align="center"><strong>用户名：</strong></div></td>
    <td width="251"><label>
      <input type="text" name="textfield">
    </label></td>
  </tr>
  <tr>
    <td><div align="center"><strong>密 码：</strong></div></td>
    <td><input type="password" name="textfield2"></td>
  </tr>
  <tr>
    <td colspan="2"><div align="center">
  <input type="submit" name="Submit" value="提交">
    
  <input type="reset" name="Submit2" value="重置">
    </div></td>
  </tr>
</table>
  <p> </p>
</body>
```

心法领悟 373：错误信息的隐藏。

如果在异步请求过程中发生了错误，那么该错误将显示在 div 区域。如果不想显示错误信息，则需要在<sx:div>标签中将 showErrorTransportText 属性设置为 false。本实例中使用 errorText 属性来定制错误信息。

实例 374　无刷新实现登录　中级

光盘位置：光盘\MR\14\374　实用指数：★★☆

实例说明

本实例实现的是无刷新登录。运行程序，当输入正确的用户名 mr 和密码 mrsoft 时，会提示登录成功信息，否则会提示错误信息，如图 14.23 所示。

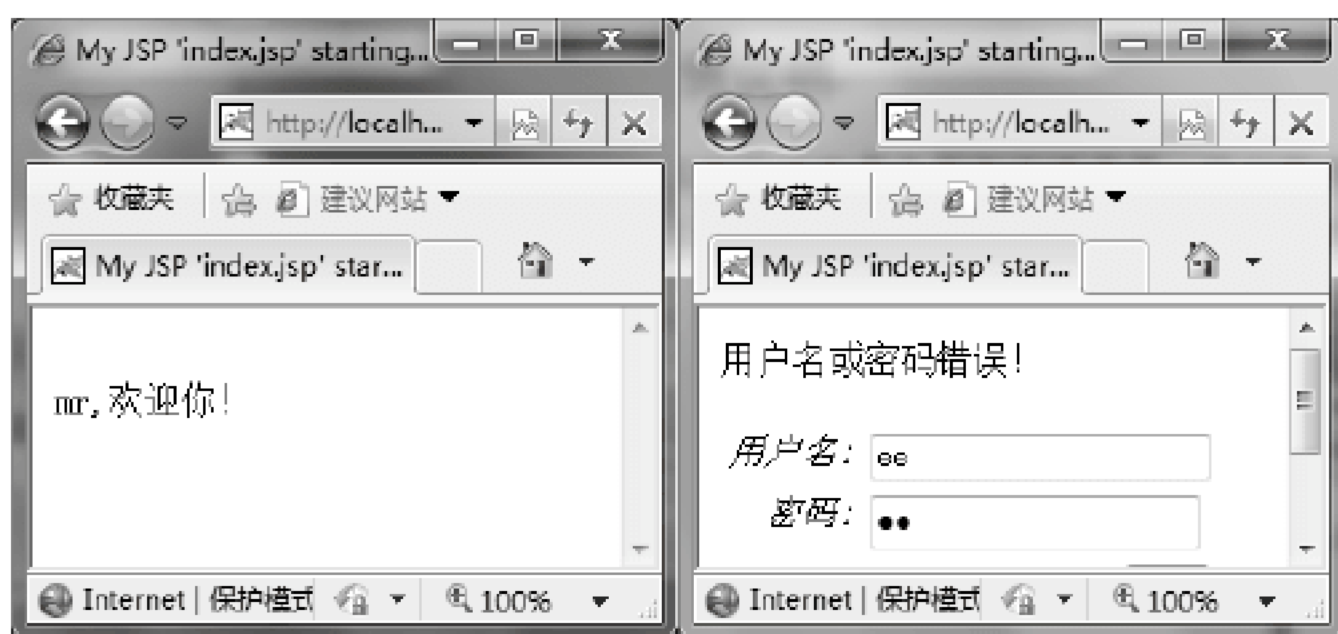


图 14.23　无刷新实现登录

关键技术

实现无刷新登录，主要是服务器端使用了 JavaScript 技术，同时把 JavaScript 代码发送到客户端并在客户端执行，页面中没有任何刷新。

（1）创建用于请求 Action 的登录页面 index.jsp，具体代码如下：

```
<body>
    <div class="div" id="logindiv">
      <div id="errordiv"></div>                                          //输出错误信息的 div
      <s:form id="formdiv" name="formdiv">
          <s:textfield name="username" label="用户名"/>
          <s:password name="password" label="密码"/>
          <s:url action="log!login" id="checklogin"></s:url>             //请求 Action
          <sx:submit value="登录" formId="formdiv"   href="%{checklogin}" targets="successdiv" executeScripts="true" />
      </s:form>
    </div>
    <div id="successdiv" class="div"></div>                              //输出正确信息的 div
</body>
```

（2）创建 Action 的流转页面 showdiv.jsp，根据用户输入内容的不同，该页面进行不同的处理。具体代码如下：

```
<body>
  <s:if test="%{#request.st=='success'}">                                //对输入的用户名进行判断
      <s:property value="#session.username"/>,欢迎你！
      <script type="text/javascript">                                    //JavaScript 代码
        document.getElementById('errordiv').innerHTML='';                //把 errordiv 中的内容设置为空
        document.getElementById('logindiv').style.display='none';        //将登录框隐藏
        document.getElementById('loginsuccessdiv').style.display='';     //将成功信息进行显示
      </script>
  </s:if>
  <s:elseif test="%{#request.st=='failed'}">                             //对输入的用户名进行判断
      <script type="text/javascript">
        document.getElementById('errordiv').innerHTML='用户名或密码错误！';  //提示错误信息
      </script>
  </s:elseif>
  <s:else>
      <script type="text/javascript">
        document.getElementById('logindiv').style.display='';            //显示登录框
        document.getElementById('loginsuccessdiv').innerHTML='';         //将成功信息置空
        document.getElementById('loginsuccessdiv').style.display='none'; //将成功信息隐藏
      </script>
  </s:else>
  </body>
```

秘笈心法

心法领悟 374：执行 JavaScript 代码。

如果服务器的响应包含 JavaScript 代码，同时客户端希望在本页内执行服务器的 JavaScript 代码，则可以在 submit 标签中使用 executeScripts="true"。

实例 375　无刷新实现注销

光盘位置：光盘\MR\14\375

中级

实用指数：★★☆

实例说明

本实例实现的是无刷新注销。运行程序，当输入正确的用户名 mr 和密码 mrsoft 时，会提示登录成功信息，同时显示“注销”超链接，单击该超链接即可注销该用户；而当输入错误的用户名或密码时会提示错误信息。本实例的运行结果如图 14.24 所示。

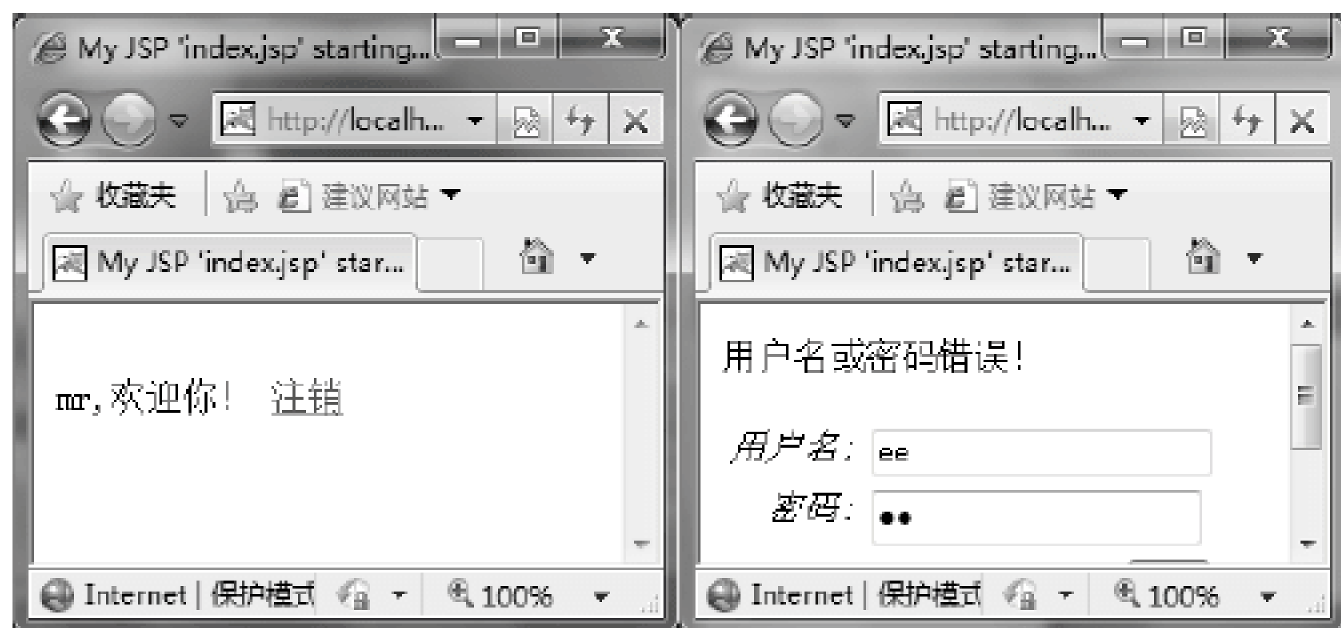


图 14.24　无刷新实现注销

关键技术

实现页面的无刷新注销，主要是通过单击“注销”超链接把放入 Session 中的数据清空；同时服务器端使用 JavaScript 技术，把 JavaScript 代码发送到客户端并在客户端执行，页面中没有任何刷新。

（1）创建用于请求 Action 的登录页面 index.jsp。具体代码如下：

```
<body>
    <div class="div" id="logindiv">
      <div id="errordiv"></div>                                    //输出错误信息的 div
      <s:form id="formdiv" name="formdiv">
        <s:textfield name="username" label="用户名"/>
        <s:password   name="password" label="密码"/>
        <s:url action="log!login" id="checklogin"></s:url>          //请求 Action
        <sx:submit value="登录" formId="formdiv"   href="%{checklogin}" targets="successdiv" executeScripts="true" />
      </s:form>
    </div>
    <div id="successdiv" class="div"></div>                         //输出正确信息的 div
</body>
```

（2）创建 Action 的流转页面 showdiv.jsp，根据用户输入内容的不同和用户单击“注销”超链接的情况，该页面将进行不同的处理。具体代码如下：

```
<s:if test="%{#request.st=='success'}">
      <s:property value="username"/>,欢迎你！
      <s:url action="log!out" id="outurl"></s:url>
      <sx:a href="%{#outurl}" executeScripts="true">注销</sx:a>        //添加注销超链接
    <script type="text/javascript">                                 //JavaScript 代码
      document.getElementById('errordiv').innerHTML='';             //将 errordiv 中的内容设置为空
      document.getElementById('logindiv').style.display='none';     //将登录框隐藏
      document.getElementById('successdiv').style.display='';       //将成功信息进行显示
    </script>
  </s:if>
  <s:elseif test="%{#request.st=='failed'}">
    <script type="text/javascript">
      document.getElementById('errordiv').innerHTML='用户名或密码错误！'; //提示错误信息
    </script>
  </s:elseif>
  <s:else>
```

```
    <script type="text/javascript">
     document.getElementById('logindiv').style.display='';                    //显示登录框
     document.getElementById('successdiv').innerHTML='';                      //将成功信息进行置空
     document.getElementById('successdiv').style.display='none';              //将成功信息隐藏
    </script>
 </s:else>
```

秘笈心法

心法领悟 375：注销超链接的生成。

超链接通过<sx:a/>生成，同时要为该标签指定 href 属性，其中的值必须是一个 Action，用于请求服务器。

实例 376　实现标签页

光盘位置：光盘\MR\14\376

中级　实用指数：★★☆

实例说明

本实例实现的是标签页。运行程序，即可显示各个版块的标签页，如图 14.25 所示。

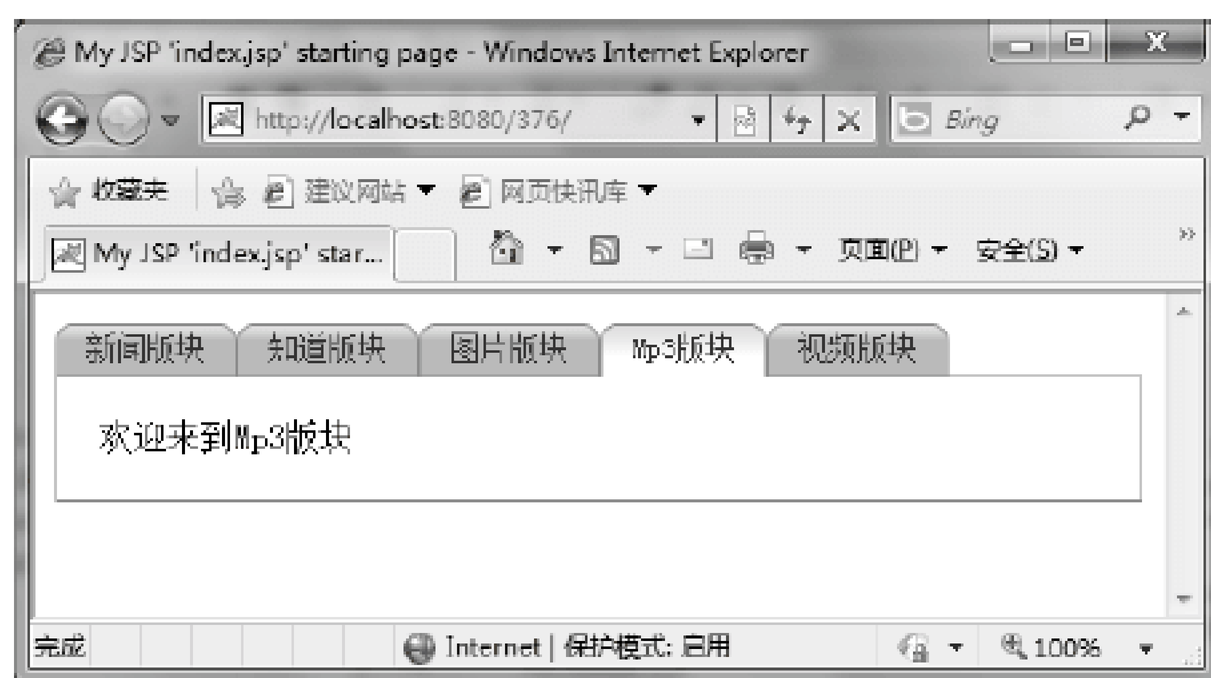


图 14.25　实现标签页

关键技术

实现标签页，要用到 tabbedpanel 标签。tabbedpanel 标签的属性分别介绍如下。

- selectedTab：指定选择哪个 Tab 页，默认选择第一个。
- closeButton：指定 Tab 页上关闭按钮的位置，可能是 tab 或 pane。
- doLayout：指定 tabbedpanel 标签是否显示固定高度，如果设置为 true，则其高度不会因 Tab 页大小的改变而改变。
- labelposition：指定 Tab 页中标签的位置。该属性有 4 个值，分别是 top、buttom、right 和 left。

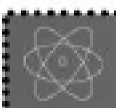
设计过程

创建用于显示标签页的 index.jsp 页面，具体代码如下：

```
<sx:tabbedpanel   id="test">
    <sx:div   id="one" label="测试 1"   cssStyle="padding:20px;">                  //设置第 1 个 Tab 页
             测试数据 1 <br>
            这是第一个测试数据
    </sx:div>
    <sx:div id="two" label="测试 2" cssStyle="padding:20px;">                      //设置第 2 个 Tab 页
              测试数据 2<br/>
            这是第二个测试数据
    </sx:div>
    <sx:div   id="three" label="测试 3"   cssStyle="padding:20px;">               //设置第 3 个 Tab 页
              测试数据 3<br/>
             这是第三个测试数据
    </sx:div>
 </sx:tabbedpanel>
```

心法领悟 376：Tab 页中的内容。

通过 tabbedpanel 标签生成的 Tab 页的内容可以是静态的，也可以是动态的。对于动态的内容，可以使用 Ajax 的方式进行加载。



实例 377　调试信息的输出

光盘位置：光盘\MR\14\377

中级

实用指数：★★☆

实例说明

本实例实现的是在登录页面添加英文版本。运行实例，即可看到在该登录页面中提供了“中文”和“英文”两个超链接。当单击“中文”超链接时，显示的是中文版本的个人登录信息；而当单击“英文”超链接时，将显示英文版本的个人登录信息。本实例的运行结果如图 14.26 所示。

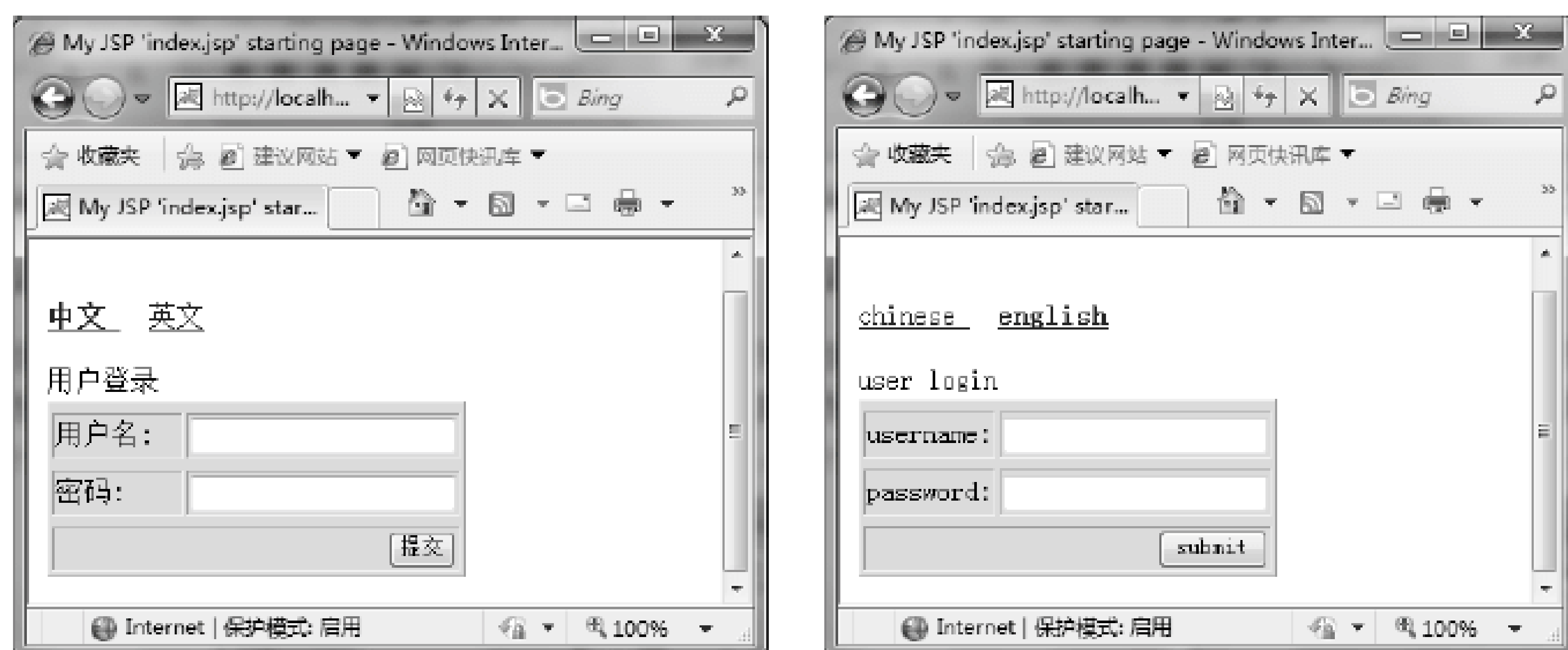


图 14.26　登录页面添加英文版本

关键技术

要实现在登录页面添加英文版本，需要设定访问用户的语言环境。具体方法为：在程序加载资源文件时，Struts2 会根据 ActionContext 的 getLocale()方法返回值加载与之相对应的资源文件；如果想改变当前的语言环境，只需将新的 Locale 对象调用 setLocale()方法，然后保存到 ActionContext 中。

设计过程

（1）创建英文版本的资源文件 RegisterAction_en.properties，具体代码如下：

```
username=username                                          //将相应的键和值一一对应
password=password
chinese=chinese
english=english
submit=submit
timu=user login
```

（2）创建中文版本的资源文件 RegisterAction_zh_CN.properties，该代码只是将对应的值改为中文，其他不变；然后创建用于显示不同版本语言的登录页面 index.jsp。具体代码如下：

```
<s:set name="current_locale" value="#session['WW_TRANS_I18N_LOCALE']==null?locale:#session['WW_TRANS_I18N_LOCALE']">
</s:set>                                                   //获取当前 locale
   <s:url id="chi" value="rege.action">
      <s:param name="request_locale" value="@java.util.Locale@CHINA"></s:param>      //创建中文和英文超链接的 URL
   </s:url>
   <s:url id="eng" value="rege.action">
      <s:param name="request_locale" value="@java.util.Locale@ENGLISH"></s:param>
   </s:url>
  <s:if test="#current_locale.equals(@java.util.Locale@CHINA)">       //对选择的超链接进行判断，然后相应地设置其字体
   <sx:a href="%{#chi}">
   <strong><s:text name="chinese"/></strong>
   </sx:a>
    
   <sx:a href="%{#eng}">
     <s:text name="english"/>
  </sx:a>
  </s:if>
  <s:else>
  <sx:a href="%{#chi}">
```

```
    <s:text name="chinese"/>
    </sx:a>
     
    <sx:a href="%{#eng}">
      <strong><s:text name="english"/></strong>
  </sx:a>
  </s:else>
<s:form>                                                                    //创建登录表单信息
  <s:text name="timu" ></s:text>
  <table width="239" border="1" bgcolor="#FFCCFF">
      <tr>
        <td width="92"><s:textfield name="username" key="username"></s:textfield></td>
      </tr>
      <tr>
        <td><s:textfield name="password" key="password"></s:textfield></td>
      </tr>
      <tr>
        <td><s:submit name="submit" key="submit"></s:submit></td>
      </tr>
  </table>
  </s:form>
```

秘笈心法

心法领悟 377：切换不同版本的语言。

无论是单击“中文”还是“英文”超链接，在该程序中都会发送一个名为 request_locale 的请求参数，参数的值为 locale 对象的字符串的值。

实例 378	数据的树状输出 光盘位置：光盘\MR\14\378	中级 实用指数：★★☆

实例说明

本实例实现的是数据的树状输出，运行程序，即可把明日科技公司各个部门的信息以树状的形式输出，如图 14.27 所示。

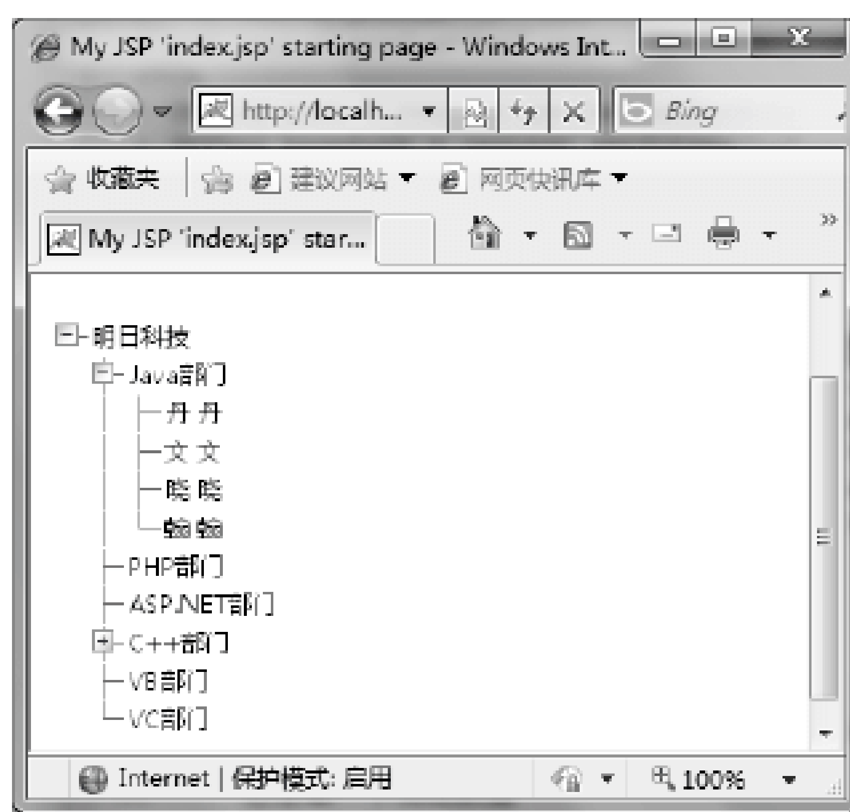


图 14.27　数据以树状输出

关键技术

实现数据的树状输出，要用到 tree 和 treenode 标签。其中，tree 用于生成一个树形结构，treenode 用于生成一个树节点。

设计过程

创建用于显示树状结构的 index.jsp 页面，具体代码如下：

```
<sx:tree id="parent" label="明日科技" showGrid="true" treeSelectedTopic="test">            //生成一个树形结构
```

```
<sx:treenode id="child1" label="Java 部门">                                           //生成一个树节点
        <sx:treenode id="sub11" label="丹 丹"/>
        <sx:treenode id="sub12" label="文 文"/>
        <sx:treenode id="sub13" label="晓 晓"/>
        <sx:treenode id="sub14" label="翰 翰"/>
 </sx:treenode>
<sx:treenode id="child2" label="PHP 部门"/>
<sx:treenode id="child3" label="ASP.NET 部门"/>
<sx:treenode id="child4" label="C++部门">
        <sx:treenode id="sub41" label="安 安"/>
        <sx:treenode id="sub42" label="华 华"/>
        <sx:treenode id="sub43" label="雪 雪"/>
        <sx:treenode id="sub44" label="鑫 鑫"/>
</sx:treenode>
<sx:treenode id="child5" label="VB 部门"/>
<sx:treenode id="child6" label="VC 部门"/>
</sx:tree>
```

秘笈心法

心法领悟 378：label 属性的作用。

tree 和 treenode 标签都有一个 label 属性，tree 标签的 label 属性用于指定根节点的标题，treenode 标签的 label 属性则用于指定树节点的标题。

实例 379	文件的树状显示 光盘位置：光盘\MR\14\379	中级 实用指数：★★☆

本实例实现的是文件的树状显示。运行程序，即可将该项目 WEB-INF 文件夹下的文件以树状的形式显示，如图 14.28 所示。

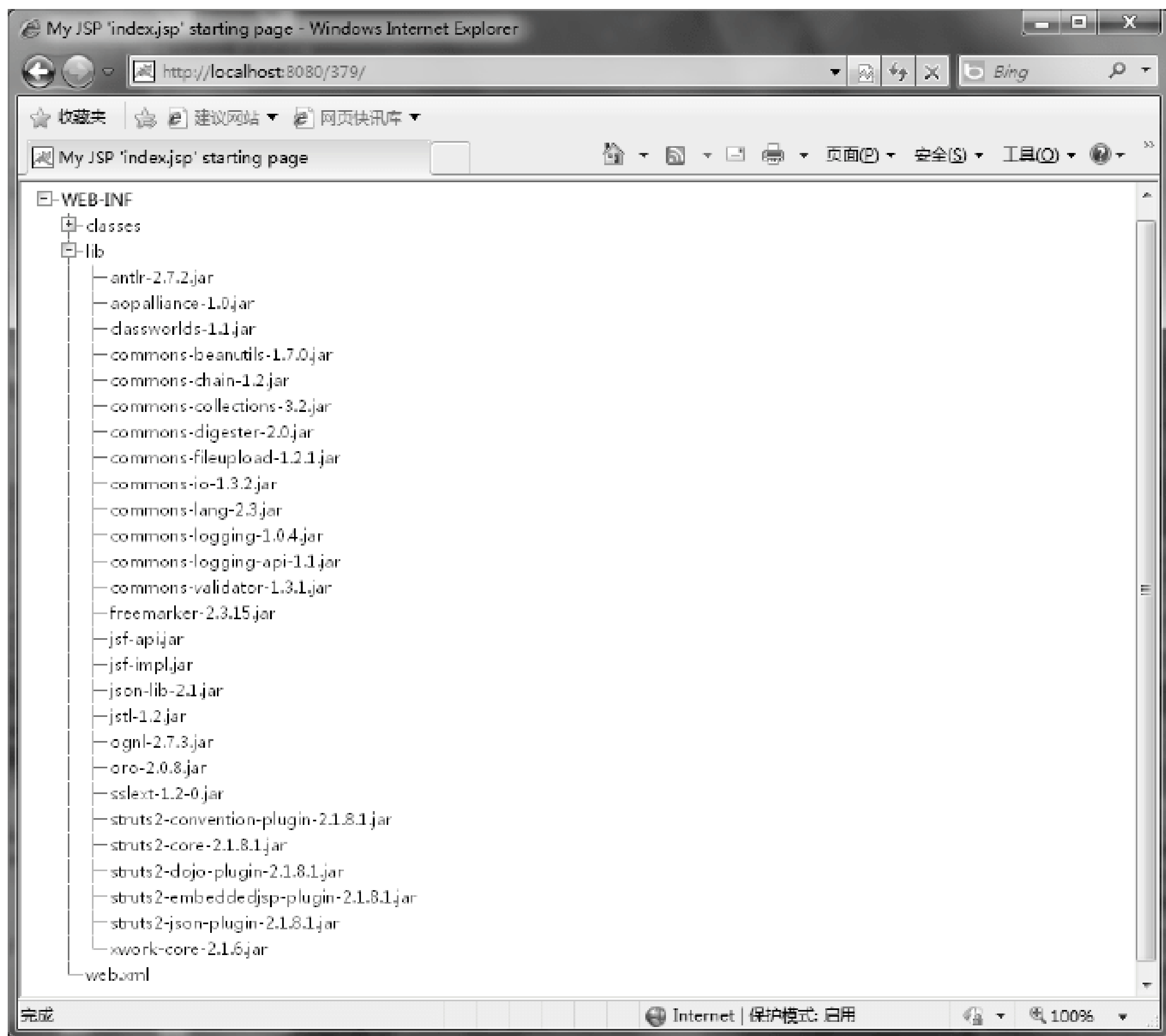


图 14.28　文件以树状显示

关键技术

要实现文件的树状显示，就要用到<sx:tree/>标签。该标签在使用树状结构的 Java 对象时，要用到下列 4 个属性。

- rootNode：根节点的对象，该属性的值可以是一个 OGNL 表达式。
- nodeIdProperty：指定用作节点 id 对象的属性。
- nodeTitleProperty：指定用作节点标题对象的属性。
- childCollectionProperty：指定用作根节点对象中一个集合类型的属性。

设计过程

（1）创建一个封装的类 FileBlock。具体代码如下：

```
private File file;
private List list=new ArrayList();
public File getFile() {
        return file;
 }
 public void setFile(File file) {
        this.file = file;
 }
 public List getList() {
        return list;
 }
 public void setList(List list) {
        this.list = list;
 }
 public FileBlock(File file)
 {
     this.file=file;
    File[] files=this.file.listFiles();
    for(int i=0;files!=null&&i<files.length;i++){
       FileBlock block=new FileBlock(files[i]);
       list.add(block);
     }
}
```

（2）使用 FileBlock 类和 tree 标签对文件进行树状显示。具体代码如下：

```
<body>
    <%
        request.setAttribute("file",new FileBlock(new File(getServletContext().getRealPath("WEB-INF"))));
    %>
  <s:set name="k" value="0"></s:set>
     <sx:tree id="root" rootNode="%{#request.file}" nodeTitleProperty="file.name" nodeIdProperty="%{#k+1}" childCollectionProperty="list" >
  </sx:tree>
 </body>
```

秘笈心法

心法领悟 379：<sx:tree/>标签中 4 个属性的进一步解释。

树的节点可以用对象表示。该对象至少需要提供两个属性，一个是节点的 id，另外一个是节点的标题。同时，对于根节点的对象，需要为其提供一个集合类型的属性，该集合包含了用作子节点的每一个对象。

实例 380	动态加载数据 光盘位置：光盘\MR\14\380	中级 实用指数：★★☆

实例说明

本实例主要实现的是在页面中动态地加载数据，并且可以通过改变代码中的数据改变显示的数据（树状），

运行结果如图 14.29 所示。

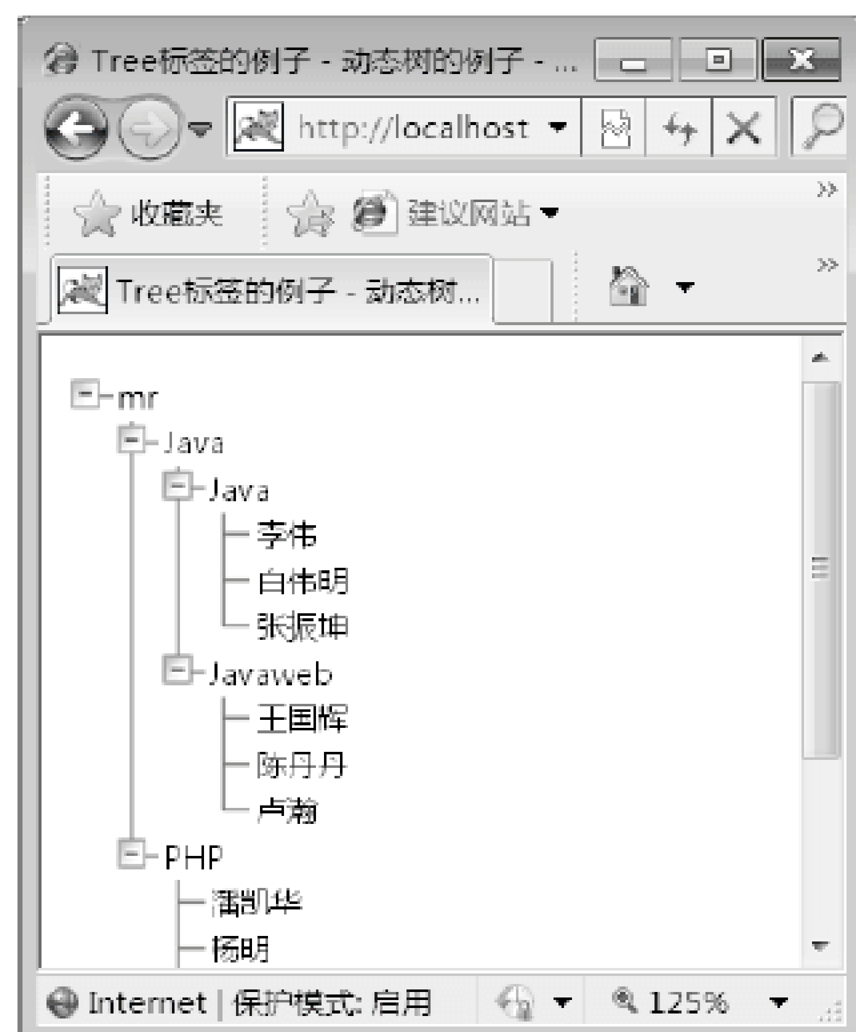


图 14.29　动态加载数据

■ 关键技术

本实例中构建动态树时要注意的是，tree 标签的 rootNode、nodeIdProperty、nodeTitleProperty 和 childCollectionProperty 这 4 个属性是必须设定的。

rootNode 属性作为树的根节点；nodeIdProperty 属性用来指定用作节点 id 的对象属性；nodeTitleProperty 属性用来指定作为节点标题的对象的属性；childCollectionProperty 属性用来指定根节点对象中的一个集合类型，而且该集合包含了子节点的对象。

■

（1）创建文件，在其中创建节点和子节点，并且给出节点数据。具体代码如下：

```
public class DateA
{
   //保存了所有节点数据的 Map
   private static Map<Long, DateA> newMap = new HashMap<Long, DateA>();

   static
   {
      new DateA(1, "mr",
            new DateA(2, "Java",
                    new DateA(3, "Java",
                            new DateA(4, "李伟"),
                            new DateA(7, "白伟明"),
                            new DateA(8, "张振坤")),
                    new DateA(9, "Javaweb",
                            new DateA(10, "王国辉"),
                            new DateA(11, "陈丹丹"),
                            new DateA(12, "卢瀚"))),
            new DateA(14, "PHP",
                    new DateA(15, "潘凯华"),
                    new DateA(16, "杨明"),
                    new DateA(17, "李慧"),
                    new DateA(18, "刘欣"),
                    new DateA(19, "张其卫")));
   }

   public static DateA getById(long id)
   {
      return newMap.get(id);
   }
   //用作节点的 id
   private long id;
   //用作节点的标题
   private String name;
   //包含子节点对象的列表
   private List<DateA> children;
   public DateA(long id, String name, DateA... children)
   {
      this.id = id;
      this.name = name;
      //初始化对象的 children 属性，并放入子对象
      this.children = new ArrayList<DateA>();
      for (DateA child : children)
      {
         this.children.add(child);
      }
      newMap.put(id, this);
```

```
    }
//省略 get()和 set()方法
}
```

（2）创建 treeExampleDynamic.jsp，在其中使用 tree 标签创建动态树，设定标签的相关属性。注意在 Dojo 中，expanded()和 collapse()方法都是被当作事件的。具体代码如下：

```
<body>
        <script type="text/javascript">
            function treeNodeSelected(message)
            {
                dojo.io.bind({
                    url: "<s:url value='/dynamicTreeSelectAction.action' />?nodeId="
                        +message.source.widgetId,
                    load: function(type, data, evt) {
                        var displayDiv = dojo.byId("displayId");
                        displayDiv.innerHTML = data;
                    },
                    mimeType: "text/html"
                });
            };
            //当用户展开节点时（单击节点左边的+）调用这个函数
            function treeNodeExpanded(message)
            {
             // alert(message.source.title + " 展开");
            }
            //当用户收缩节点时（单击节点左边的-）调用这个函数
            function treeNodeCollapsed(message)
            {
             // alert(message.source.title + " 收缩");
            }
            dojo.addOnLoad(function()
            {
                    var tree = dojo.widget.byId("parentId");
                    dojo.event.topic.subscribe(tree.eventNames.expand,treeNodeExpanded);
                    dojo.event.topic.subscribe(tree.eventNames.collapse,treeNodeCollapsed);
                    var selector = tree.selector;
                    dojo.event.connect(selector,"select","treeNodeSelected");
            });
        </script>
        <div style="float: left; margin-right: 50px;">
            <sx:tree id="parentId" rootNode="%{treeRootNode}"
                childCollectionProperty="children" nodeIdProperty="id"
                nodeTitleProperty="name">
            </sx:tree>
        </div>
        <br/><br/>
        <div id="displayId">初始化内容</div>
    </body>
```

心法领悟 380：数据的来源。

本实例为了演示的方便，将数据直接写进了代码中，读者可以试着将数据通过 JSP 页面进行传入或者使用数据库中的数据。

第15章

Struts2 框架标签应用

- OGNL 语言
- 控制标签
- 数据标签
- 表单标签

15.1 OGNL 语言

实例 381 访问 OGNL 上下文

光盘位置：光盘\MR\15\381

中级

实用指数：★★★☆

实例说明

本实例实现的是对地址栏中的参数进行输出，也就是在地址栏中手动填写参数名和参数值，执行后在页面中显示手动填写的参数值。实例运行结果如图 15.1 所示。

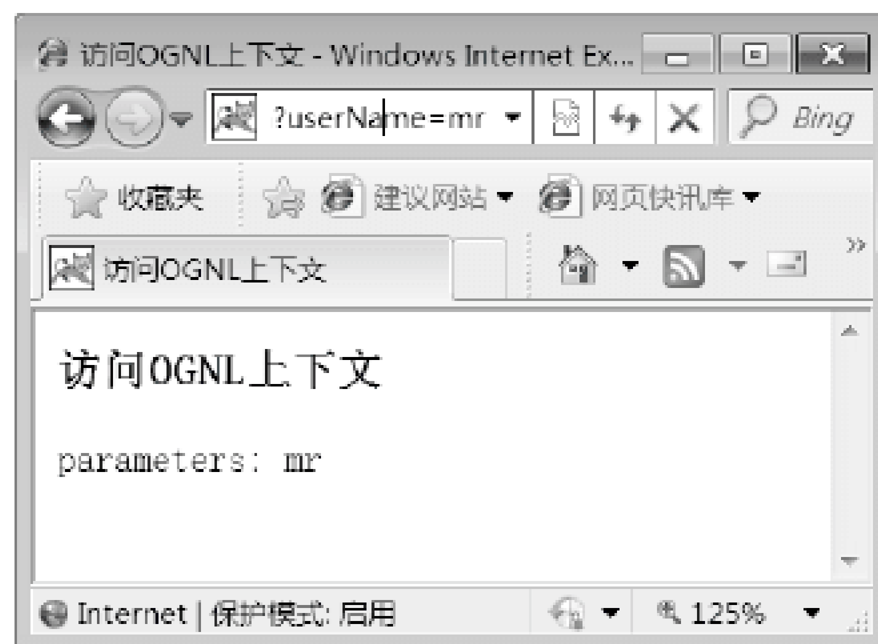


图 15.1　访问 OGNL 上下文

关键技术

OGNL 上下文中包含两个对象，例如，分别为 User 对象与 Book 对象，如果获取 User 对象与 Book 对象中的 name 属性，可以使用以下表达式。

```
#user.name
#book.name
```

上述代码相当于调用了 User 对象与 Book 对象的 getName()方法。由于 User 对象为 OGNL 上下文的根，可以将表达式中的“#”去除。例如：

```
user.name
```

设计过程

（1）创建 OgnlAction.action 文件，在其中定义相应的变量。

（2）创建简单的 JSP 页面，在其中使用“#”获取 parameters.userName。具体代码如下：

```
<body>
  <h4>访问 OGNL 上下文</h4>
    <p>parameters: <s:property value="#parameters.userName" /></p>
  </body>
```

心法领悟 381：什么是 OGNL，有什么特点。

OGNL（Object-Graph Navigation Language，对象图形化导航语言），是一种可以方便地操作对象属性的开源表达式语言。OGNL 具有如下特点：

- 支持对象方法调用，形式如 objName.methodName()。
- 支持类的静态方法调用和值访问，表达式的格式为@[类全名（包括包路径）]@[方法名|值名]。例如，@java.lang.String@format('foo %s', 'bar')或@tutorial.MyConstant@APP_NAME。
- 支持赋值操作和表达式串联。例如，price=100, discount=0.8, calculatePrice()，此表达式将返回 80。
- 访问 OGNL 上下文（OGNL context）和 ActionContext。
- 操作集合对象。

实例 382　访问 ActionContext 资源

光盘位置：光盘\MR\15\382

中级

实用指数：★★

实例说明

本实例主要是实现对 ActionContext 的资源进行访问，将一些资源属性输出到页面进行显示。实例运行结果如图 15.2 所示。

Struts2 框架在 OGNL 的基础上提高了对数据的访问能力。在 Struts2 框架中，OGNL 上下文作用于 Struts2 中的 ActionContext 对象。ActionContext 对象是 Struts2 框架中的一个核心对象，其结构如图 15.3 所示。

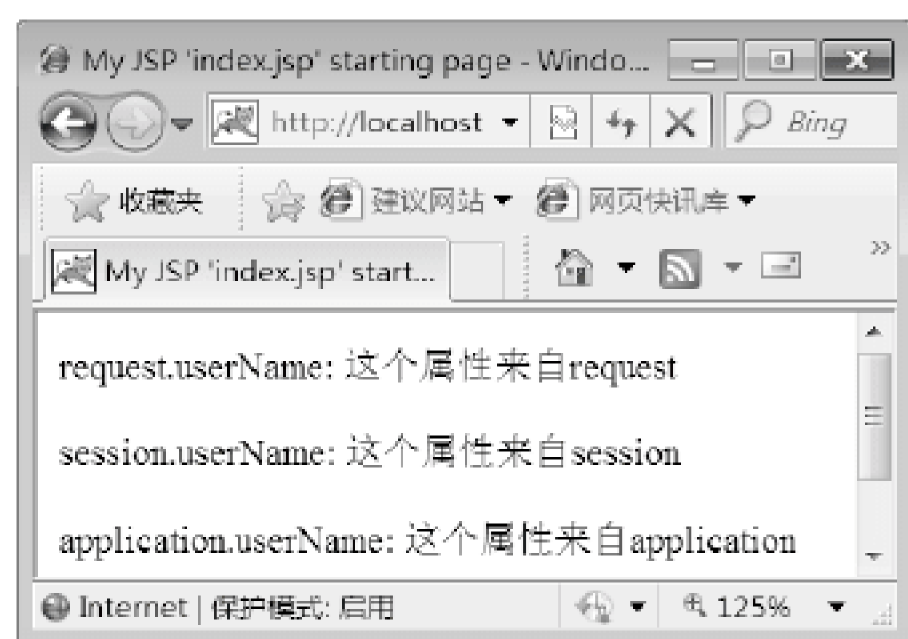


图 15.2　访问 ActionContext 资源

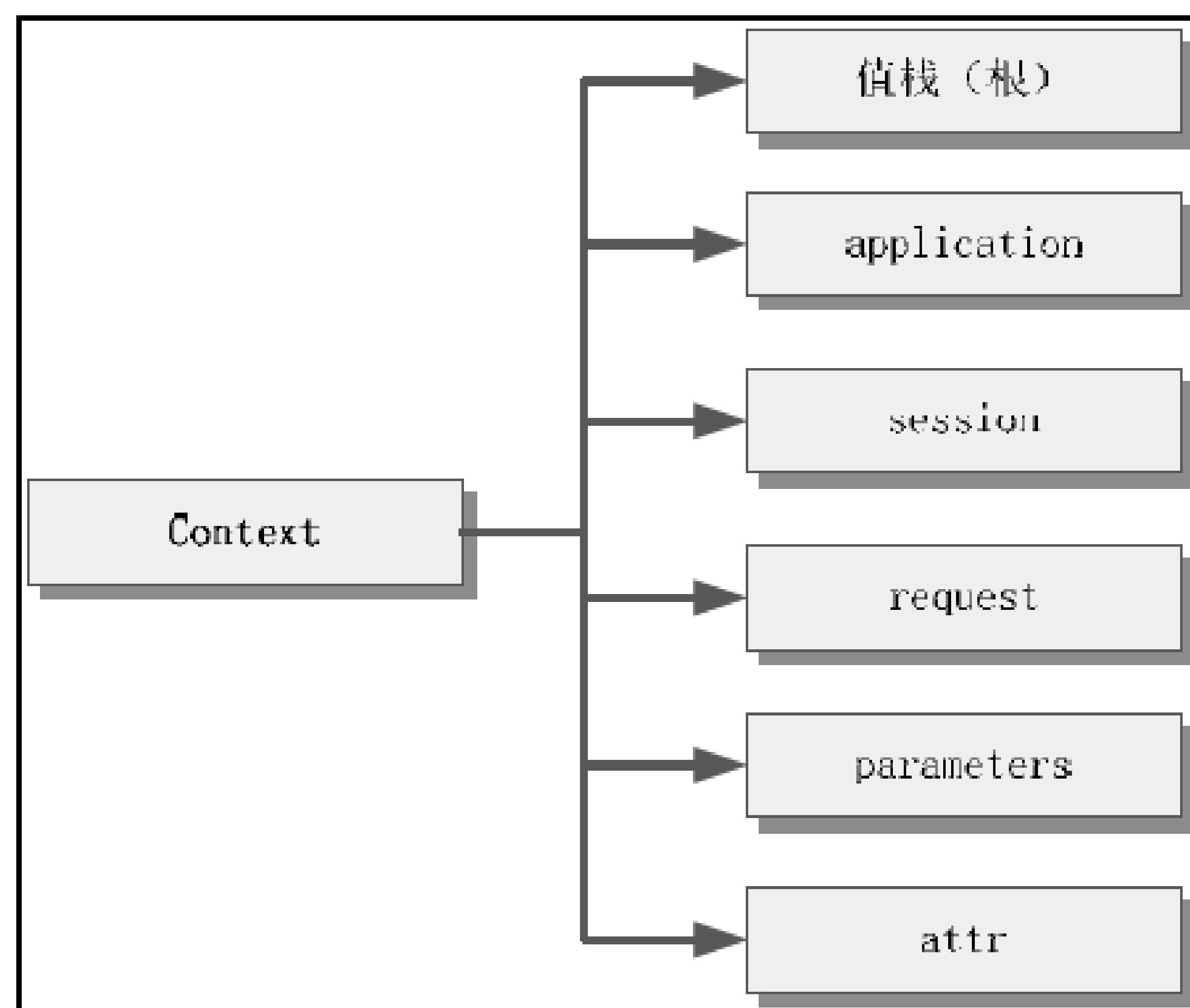


图 15.3　ActionContext 中的对象

值栈符合栈的基本特性，对象在栈中的存放是后进先出的，由 Struts2 API 中的 com.opensymphony.xwork2.ognl.OgnlValueStack 类实现。在 ActionContext 中包含多个对象，值栈是 OGNL 上下文的根，值栈中的对象可以直接调用。

在 Struts2 框架中，当接受一个 Action 请求时，Struts2 框架会创建 ActionContext 对象并实例化值栈等对象，由于 OGNL 上下文作用于 ActionContext 对象，所以通过 OGNL 表达式可以获取 ActionContext 中的所有对象。

（1）获取值栈中的对象

ONGL 上下文中的根是可以直接获取的，所以对于值栈中的对象而言，可以直接获取，原因在于 Struts2 框架中值栈是 ONGL 上下文的根。其取值方法如下：

```
${user.name}
```

（2）获取 application 中的对象

在 ActionContext 对象中包含 application，由于它并不是 ONGL 上下文的根，其取值方法如下：

```
#application.name
```

或

```
#application.['name']
```

上述代码相当于调用了 application.getAttribute("name")方法。

（3）获取 request 中的对象

与 application 相同，request 也不是 ONGL 上下文的根，其取值方法如下：

```
#request.name
```

或

```
#request.['name']
```

上述代码相当于调用了 request.getAttribute("name")方法。

（4）获取 session 中的对象

session 也不是 OGNL 上下文的根，其取值方法与 application、request 相同。

```
#session.name
```

或

```
#session.['name']
```

上述代码相当于调用了 session.getAttribute("name")方法。

（5）获取 parameters 中的值

在 Struts2 框架中，通过 OGNL 获取请求参数。其获取参数值的方式如下：

```
#parameters.name
```

或

```
#parameters.['name']
```

上述代码相当于调用了 request.getParameter("name")方法。

（6）获取 attr 中的值

如果不指定范围，可以使用 attr 来获取属性值，它将按照 page、request、session、application 的次序进行搜索。attr 的使用方法如下：

```
#attr.name
```

或

```
#attr.['name']
```

设计过程

（1）创建 OgnlAction.action 文件，在其中定义变量和编写 get()、set()方法。

（2）创建 index.jsp 文件，应用“#”获取相应资源的属性。具体代码如下：

```
<body>
    //使用“#”获取属性
    <p>request.userName: <s:property value="#request.userName" /> </p>
    <p>session.userName: <s:property value="#session.userName" /> </p>
    <p>application.userName: <s:property value="#application.userName" /> </p>
    <p>attr.userName: <s:property value="#attr.userName" /> </p>
  </body>}
```

秘笈心法

心法领悟 382：访问静态方法与属性。

Struts2 框架中的 OGNL 表达式支持对象的静态方法与静态属性的调用，类似于 Java 语言中的静态引入，需要使用字符“@”进行标注。调用静态属性的方法如下：

```
@com.lyq.bean.Bean@NAME
```

上述代码相当于调用了 Bean.NAME 静态属性。

与调用静态属性的方法相同，调用静态方法的格式如下：

```
@com.lyq.bean.Bean@greeting()
```

在 Struts2 框架中提供了一个是否允许 OGNL 调用静态方法的常量，即 struts.ognl.allowStaticMethodAccess。其默认属性 false，也就是说，在默认情况下，Struts2 不允许 OGNL 调用静态方法。因此，如果开发中需要使用 OGNL 调用静态方法，必须将此常量的值设置为 true；否则，将无法通过 OGNL 调用对象静态方法。要更改这个常量的值，可以在 struts.xml 配置文件中加入如下代码进行更改。

```
<constant name="struts.ognl.allowStaticMethodAccess" value="true"/>
```

实例 383	用“#”过滤筛选集合 光盘位置：光盘\MR\15\383	中级 实用指数：★★☆

实例说明

本实例将对图书的价格按照 50 美元为标准进行过滤，并对某一特定名称的图书的价格进行显示，运行结果

如图 15.4 所示。

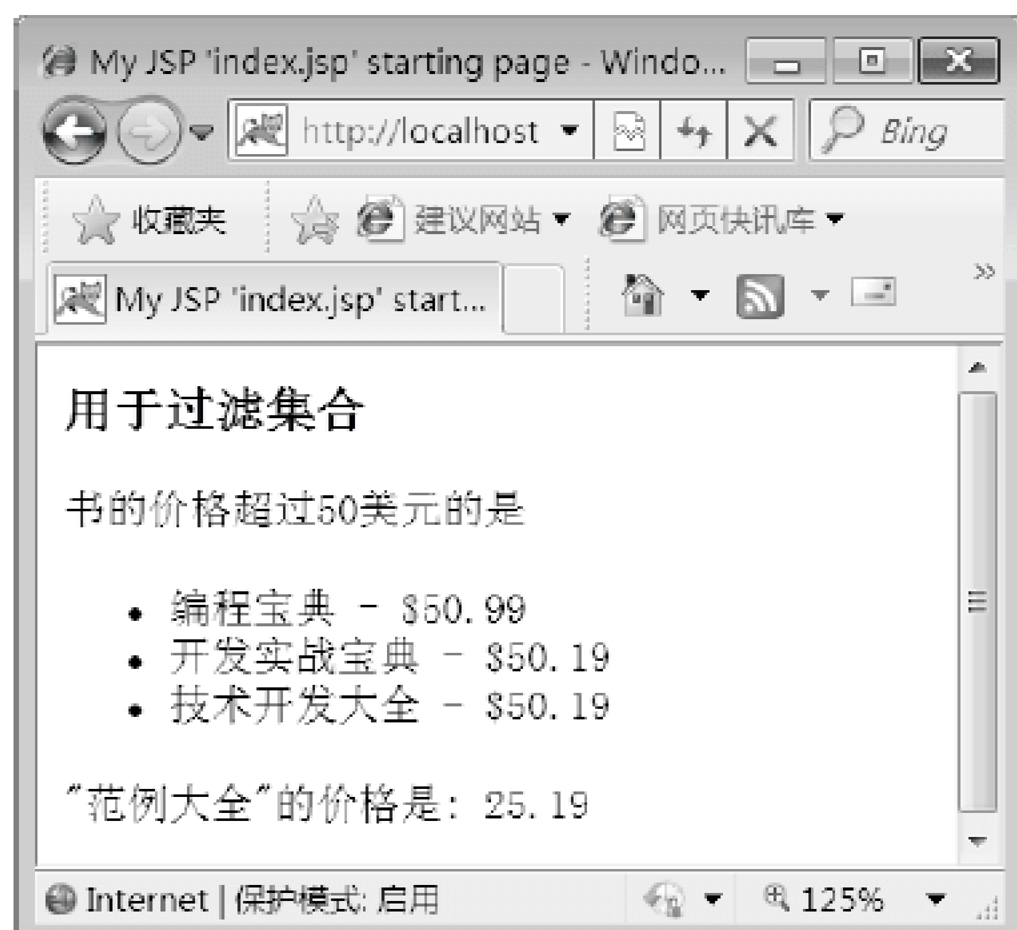


图 15.4　用“#”过滤集合

关键技术

本实例主要用到 OGNL 语言对集合的选择操作。所谓选择，就是通过一定的条件获取集合中满足这一条件的数据（主要针对的是行）。例如，在一个集合中包含多个学生对象，获取这个集合中年龄大于 10 的所有学生的信息，就是选择操作。在 OGNL 表达式中可以通过如下代码实现：

```
list.{?#this.age > 10}
```

设计过程

（1）创建 OgnlAction 文件，在其中定义变量，并通过 add()方法对变量进行赋值。具体代码如下：

```
public class OgnlAction extends ActionSupport {
        //定义 List 类型 books 变量
        private List<Book> books;
        public List<Book> getBooks() {
             return books;
        }
             public String execute() {
             //初始化 books 对象
             books = new LinkedList<Book>();
             //应用 add()方法为 List 类型 books 变量赋值
             books.add(new Book("978-0735619678", "编程宝典", 50.99));
             books.add(new Book("978-0596007867", "典型模块大全", 35.96));
             books.add(new Book("978-0201633610", "开发实战宝典", 50.19));
             books.add(new Book("978-0596527341", "范例大全", 25.19));
             books.add(new Book("978-0735605350", "技术开发大全", 50.19));
             return SUCCESS;
      }
  }
```

（2）创建 JSP 文件，使用“#”对集合进行过滤选择。具体代码如下：

```
<body>
      <h3>用于过滤集合</h3>
      <p>书的价格超过 50 美元的是 </p>
      <ul>
            <s:iterator value="books.{?#this.price > 50}">
                  <li><s:property value="title" /> - $<s:property value="price" /></li>
            </s:iterator>
      </ul>
      <p> "范例大全"的价格是：<s:property value="books.{?#this.title=='范例大全'}.{price}[0]"/></p>
</body>}
```


心法领悟 383：OGNL 表达式中的选择操作符及其说明。

OGNL 表达式中的选择操作符及其说明如表 15.1 所示。

表 15.1　OGNL 表达式中的选择操作符及其说明

符　　号	说　　明
?	获取满足指定条件的所有元素
^	获取满足指定条件的所有元素中的第一个元素
$	获取满足指定条件的所有元素中的最后一个元素



实例 384　用“#”构造 Map

光盘位置：光盘\MR\15\384

中级　实用指数：★★★★☆

实例说明

本实例将使用 OGNL 语言创建一个集合对象，并将集合对象中的元素进行输出。实例运行结果如图 15.5 所示。

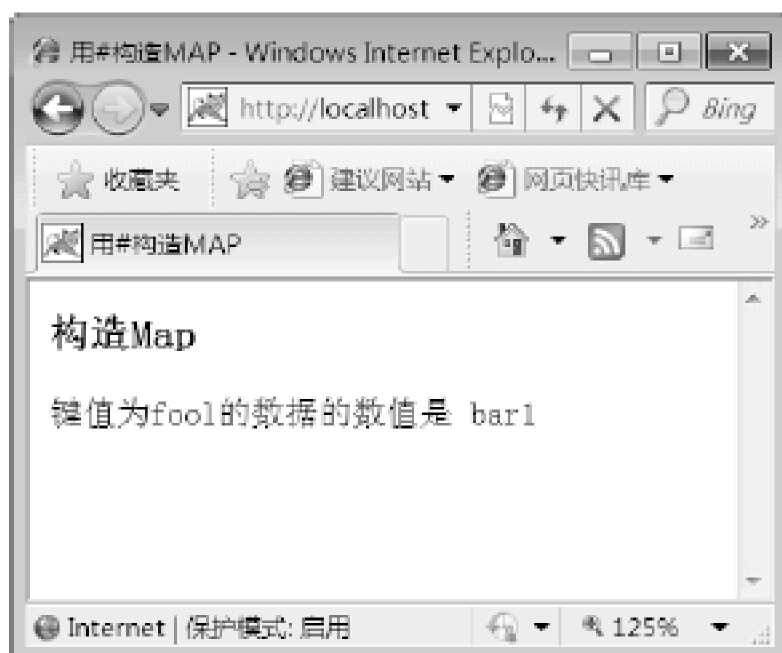


图 15.5　用“#”构造 Map

关键技术

本实例的实现主要是应用了 OGNL 语言可以对集合进行创建和操作这一特性。下面来看一下创建 Map 集合的语法格式。

```
#{'key1':'value1', 'key2':'value2'…'keyN': 'valueN'}
```

Map 元素通过 key 来访问。

设计过程

在项目中创建 JSP 文件，在其中引用 OGNL 语言，与 set 标签结合使用，进行参数的设定。具体代码如下：

```
<body>
    <h3>构造 Map</h3>
    <s:set name="foobar" value="#{'foo1':'bar1', 'foo2':'bar2'}" />
    <p>键值为 fool 的数据的数值是 <s:property value="#foobar['foo1']" /></p>
</body>}
```

秘笈心法

心法领悟 384：不同标签对 OGNL 语言的使用。

这里要说明一下，Struts2 中不同的标签对 OGNL 表达式的理解是不一样的。当某些标签“看不懂”类似 #myMap['key1']的语句时，可以用“%{}”将其括进去，“翻译”一下。



实例 385　获取 Request 的 account 属性

光盘位置：光盘\MR\15\385

中级　实用指数：★★★

实例说明

本实例将实现银行转账模拟系统，首先输入转出方、转入方账号以及金额，然后单击“提交”按钮，即可

显示相应的输入信息，以便用户确认。实例运行结果如图 15.6 和图 15.7 所示。



图 15.6　银行转账模拟系统

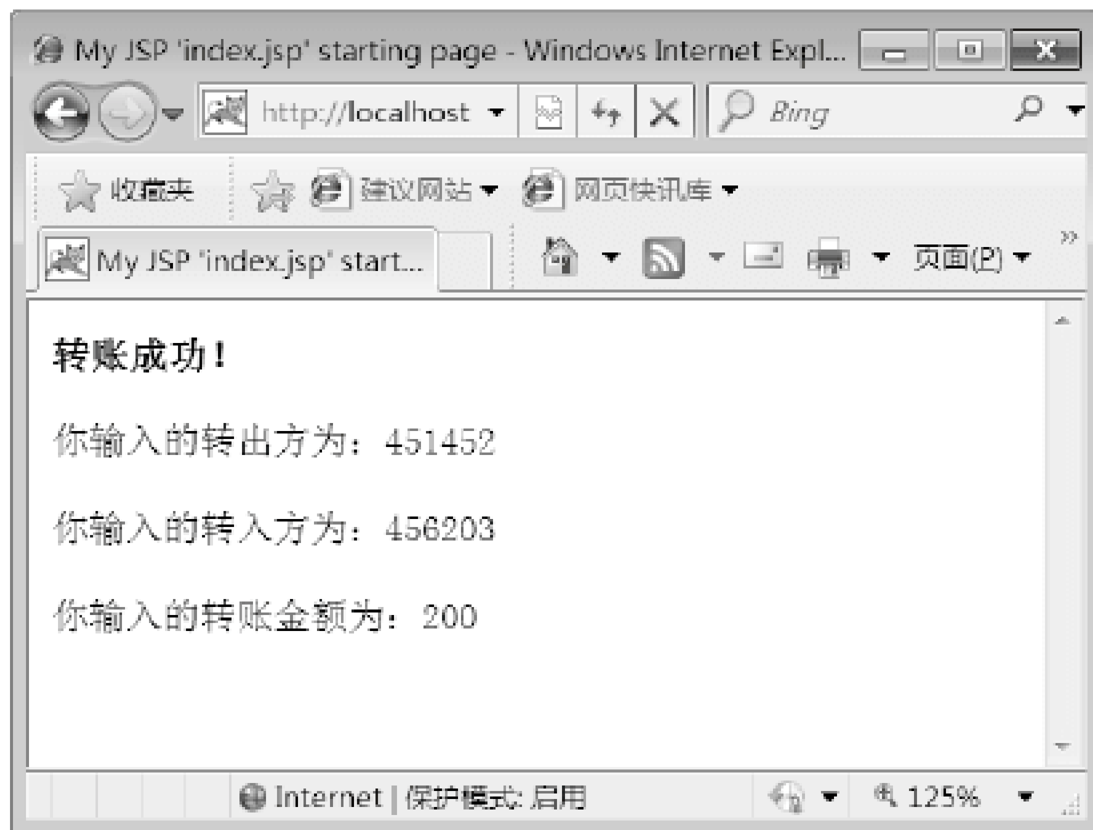


图 15.7　成功页面

关键技术

本实例的实现主要是应用了“#”的#request 属性。在 OGNL 语言中，#request.account 相当于 JSP 中的 request.getAttribute("account")。

设计过程

（1）创建提交页面 index.jsp，在其中编写相应的提交表单。

（2）创建显示页面 success.jsp，在其中应用#request.account 来获取提交的数据。具体代码如下：

```
<body>
<p class="STYLE1">转账成功！</p>
<p>你输入的转出方为：<s:property value="#request.from"/></p>
<p>你输入的转入方为：<s:property value="#request.to"/></p>
你输入的转账金额为：<s:property value="#request.account"/>
</body>
```

秘笈心法

心法领悟 385：页面数据的获取。

对于网页中的数据，在获取时并不一定要使用类似 JSP 中的 request.getAtttibute()等代码，完全可以使用 OGNL 语言中的一些属性或者 Struts2 等提供的一些标签，这样可以很大程度地提高开发效率和代码的可读性。

实例 386	在资源文件中引用 OGNL 光盘位置：光盘\MR\15\386	中级 实用指数：★★★★

实例说明

本实例主要是实现在资源文件中使用 OGNL 获取参数，并且在页面中进行显示。实例运行结果如图 15.8 所示。

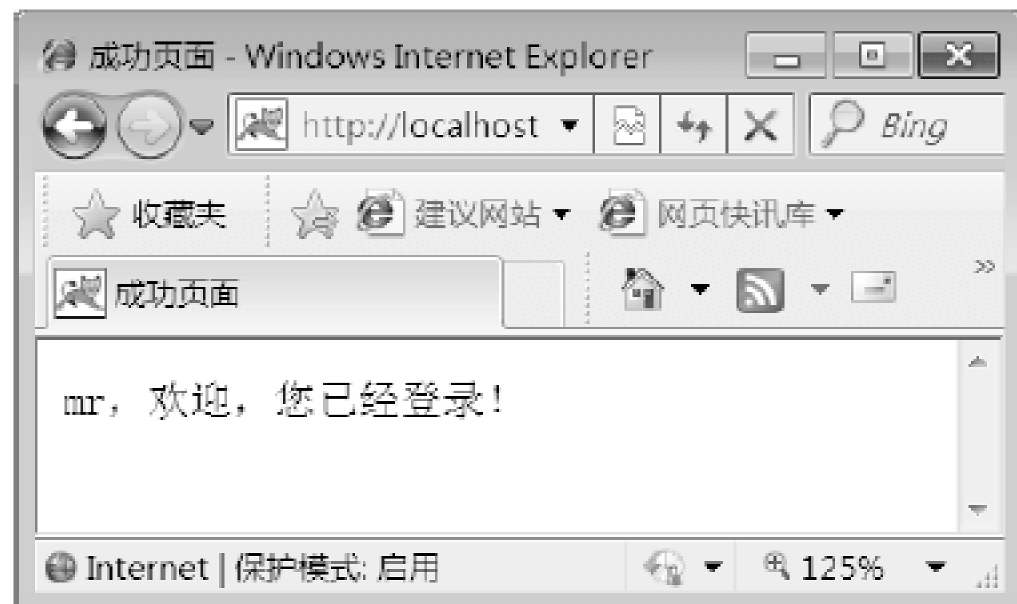


图 15.8　引用 OGNL

关键技术

在资源文件中使用 OGNL 语言的语法如下（以获取用户名为例）：

```
succTip=${username},欢迎您，您已登录！
```

在登录成功页面使用如下语法进行获取：

```
<s:text name="succTip"></s:text>
```

设计过程

（1）创建资源文件 messageResource_zh_CN.properties，在其中使用 OGNL 语言。关键代码如下：

```
succTip=${username},欢迎您，您已登录！
```

（2）创建 JSP 文件 welcome.jsp，在其中使用<s:text>标签对数据进行获取。具体代码如下：

```
<html>
    <head>
        <title><s:text name="succPage"/></title>
    </head>
    <body>
        ${requestScope.tip}<br/>
        <s:text name="succTip"></s:text>
    </body>
</html>
```

秘笈心法

心法领悟 386：在资源文件中使用 OGNL 语言。

在资源文件中直接使用 OGNL 语言，可以更好地降低程序的耦合度。如果想改动获取的数据，只需要在资源文件中进行修改，程序的可维护性就大大提高了。

实例 387 在 struts.xml 中引用 OGNL

光盘位置：光盘\MR\15\387

高级

实用指数：★★★

实例说明

本实例主要是实现用户的注册。在注册后，系统将对用户输入的注册信息与给定的用户名进行比对，判断注册的用户名是否可以幸运地成为高级会员。实例运行结果如图 15.9 和图 15.10 所示。

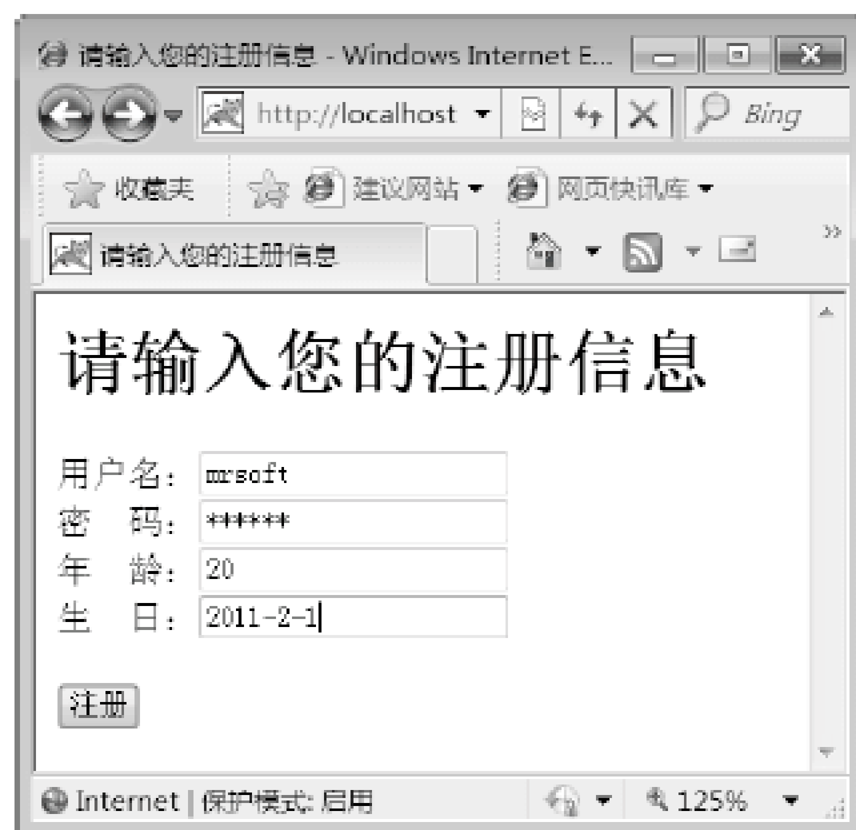


图 15.9　注册页面

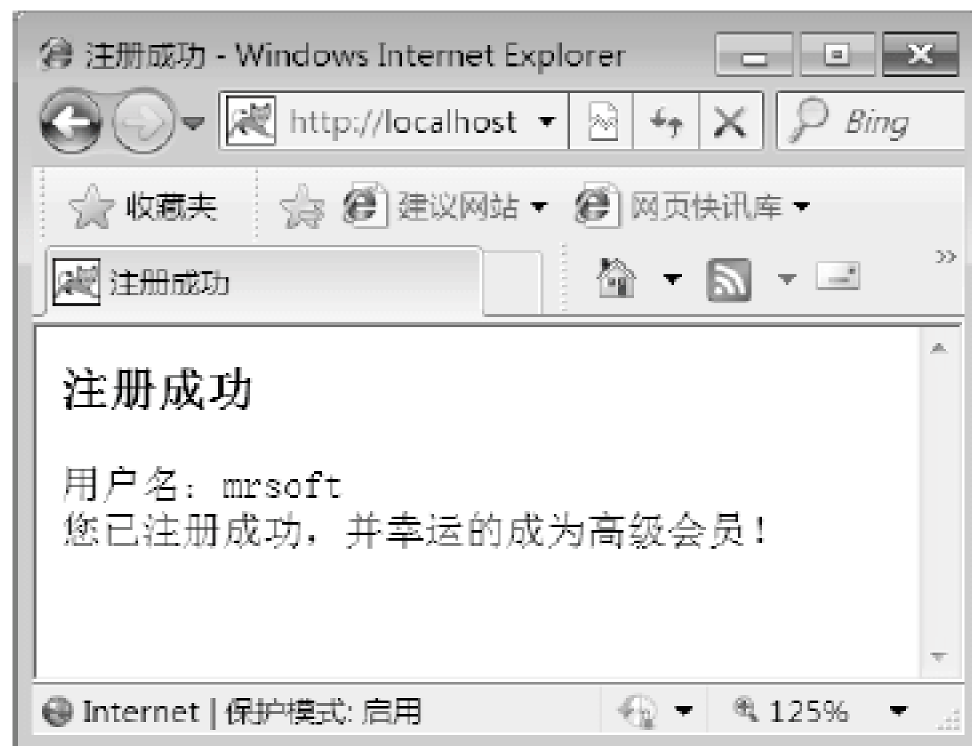


图 15.10　信息显示页面

关键技术

实现 struts.xml 中的 OGNL 语言的应用，具体代码如下：

```
<action name="regist" class="lee.RegistAction">
            <result name="input">/regist.jsp</result>
```

```
            <result type="redirect">equl.action?name=${name}</result>
</action>
```

设计过程

（1）创建两个普通的 Action 文件，在其中定义相应的变量并编写 get()、set()方法。

（2）在 struts.xml 中引用 OGNL 语言，实现参数的传递。具体代码如下：

```
<package name="lee" extends="struts-default">
         <action name="equl" class="lee.EqulAction">
           <result name="success1">/show1.jsp</result>
         </action>
        <action name="regist" class="lee.RegistAction">
            <result name="input">/regist.jsp</result>
            //使用 OGNL 语言实现参数的传递和重定向
            <result type="redirect">equl.action?name=${name}</result>
        </action>
    </package>
```

秘笈心法

心法领悟 387：在 OGNL 语言中操作普通的属性与方法。

在 Struts2 框架中使用 OGNL 表达式语言，需要借助 Struts2 标签输出。例如，获取 User 对象中的 name 属性，在 JSP 页面中可以使用如下代码获取。

```
<s:property value="user.id"/>
```

在 OGNL 表达式中，获取属性的方法主要有两种，除了上面介绍的方法外，也可以通过下面的代码获取。

```
<s:property value="user[id]"/>
```

OGNL 不仅支持属性的调用，也支持方法的调用。在 JSP 页面中，可以使用如下代码调用 User 类中的方法。

```
<s:property value="user.say()"/>
```

15.2　控 制 标 签

实例 388	判断用户是否存在 光盘位置：光盘\MR\15\388	中级 实用指数：★★☆

实例说明

本实例就是一个简单的对于用户是否存在的判断。运行程序时，数据要通过地址栏进行参数的传入。实例运行结果如图 15.11 所示。

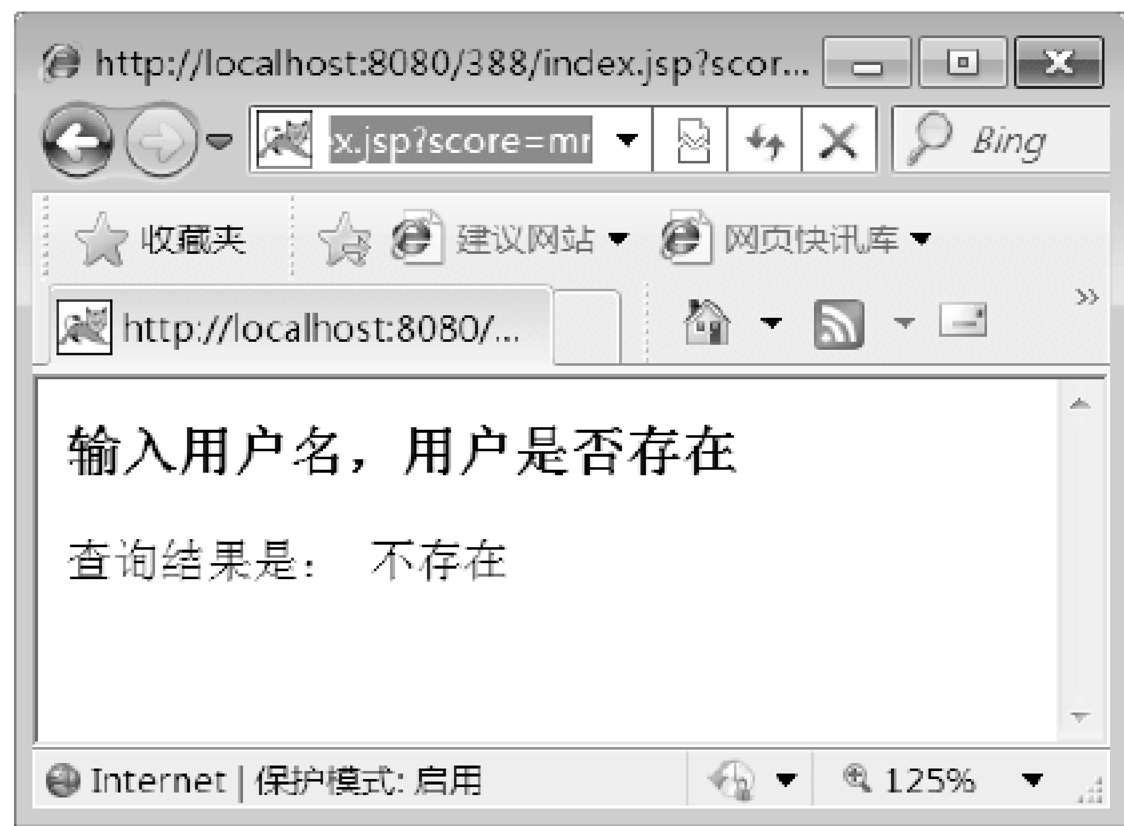


图 15.11　用户判断页面

if 标签是 Struts2 框架提供的一个流程控制标签，针对某一逻辑的多种条件进行处理，通常表现为“如果满

足某种条件，就进行某种处理，否则进行另一种处理”。与 Java 语言相同，Struts2 框架的标签同样支持 if、else if、else 语句判断，其标签如下。

- <s:if>：基本流程控制标签，用于在满足某个条件的情况下，执行标签体中的内容。<s:if>标签可以单独使用。
- <s:elseif>：此标签需要与<s:if>标签配合使用，用于在不满足<s:if>标签中条件的情况下判断是否满足<s:elseif>标签中的条件，如果满足此标签中的条件，那么将执行<s:elseif>标签体中的内容。
- <s:else>：此标签需要与<s:if>标签或<s:elseif>标签配合使用，在不满足所有条件的情况下，可以使用<s:else>标签来执行此标签中的内容。

具体语法格式如下：

```
<s:if test="表达式(布尔值)">
    输出结果...
</s:if>
<s:elseif test="表达式(布尔值)">
    输出结果...
</s:elseif>
可以使用多个<s:elseif>
…
<s:else>
    输出结果...
</s:else>
```

设计过程

创建 index.jsp 文件，在其中应用控制判断标签。具体代码如下：

```
<h3>输入用户名，用户是否存在</h3>
<s:set name="score" value="#parameters.score[0]"/>
查询结果是：
<s:if test="#score==mrs">
      存在
</s:if>
<s:elseif test="#score==mrsoft">
      存在
</s:elseif>
<s:else>
      不存在
</s:else>
```

秘笈心法

心法领悟 388：使用 if 标签和 else 标签的注意事项。

<s:if>标签与<s:elseif>标签都用于在满足某种条件的情况下执行指定标签体内的内容，所以使用<s:if>标签与<s:elseif>标签需要为其指定一定的条件表达式，而<s:else>标签是在不满足所有条件的情况下执行标签体内的内容，因此使用<s:elseif>标签不需要为其指定条件表达式。



实例 389	用户不存在的提示 光盘位置：光盘\MR\15\389	中级 实用指数：★★★

发帖主要是为了讨论相关话题，这也是论坛系统中的主要功能。通常情况下，需要在论坛中注册一个用户名，然后成功登录，才能在论坛中发帖。本实例将实现用户登录后进行发帖，运行结果如图 15.12 所示。



图 15.12　用户不存在的提示

关键技术

在发帖模块中用到的主要技术有：首先使用 Validator 框架验证表单，然后在 Action 类中获取表单数据，最后通过 SQL 语句向数据表中插入记录。相信读者对后两者的使用都已经非常熟悉了，这里不再介绍，下面简单介绍一下如何使用 Validator 框架验证表单。

应用 Validator 框架进行验证需要注意以下两点：

- 在 Struts 中应用 Validator 验证框架进行表单验证时，与 form 表单对应的 ActionForm Bean 不能继承标准的 org.apache.struts.action.ActionForm 类，通常情况下应继承 org.apache.struts.validator.ValidatorForm 类。
- 应用 Validator 框架需要将 jakarta-oro.jar 和 commons-validator.jar 两个文件复制到应用的 WEB-INF\lib 目录下。

设计过程

（1）首先创建一个 IfAction 文件，在其中定义变量和相应的处理方法。

（2）创建 JSP 文件，编写表单代码并且应用控制判断标签。具体代码如下：

```
<body>
    <s:form action="if">
    <s:textfield name="name" label="姓名"></s:textfield>
    <s:submit value="提交"></s:submit>
    </s:form>
    <s:if test="name=='mr'">
  欢迎您，mr！
    </s:if>
    <s:elseif test="name =='mrsoft'">
  请确认您的用户名再进入！
    </s:elseif>
    <s:else></s:else>
 </body>
```

秘笈心法

心法领悟 389：判断标签的扩展应用。

控制标签中的判断标签不一定只是判断本页面中的数据，也可以通过 Action 等进行传入，这样就大大提高了控制判断标签的应用范围。

实例 390	简单的计算器 光盘位置：光盘\MR\15\390	中级 实用指数：★★☆

实例说明

本实例主要是实现简单的计算器，通过下拉列表选择计算的类型。实例的运行结果如图 15.13 和图 15.14 所示。

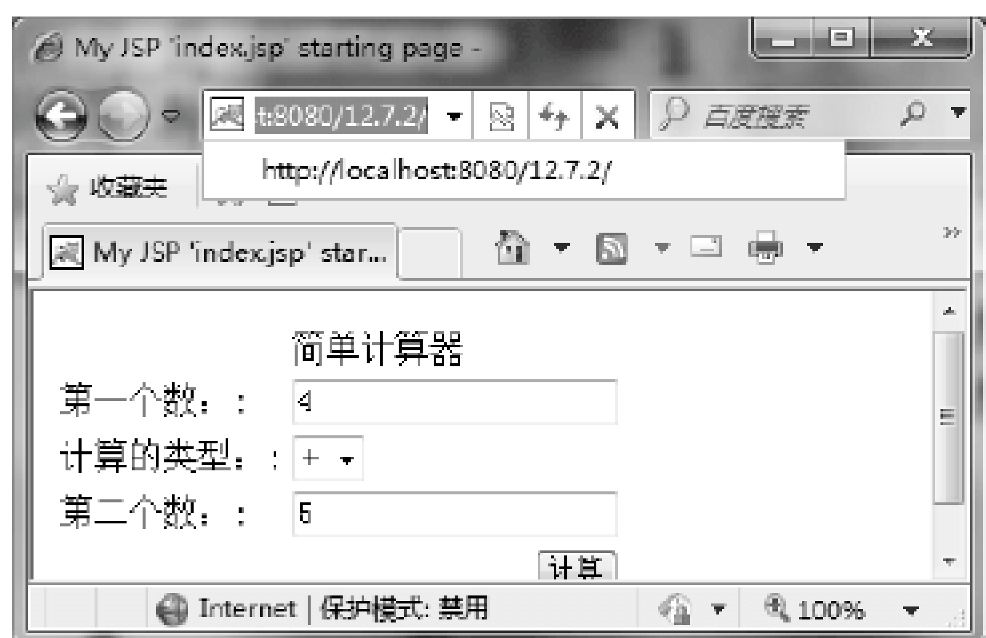


图 15.13　计算选择页

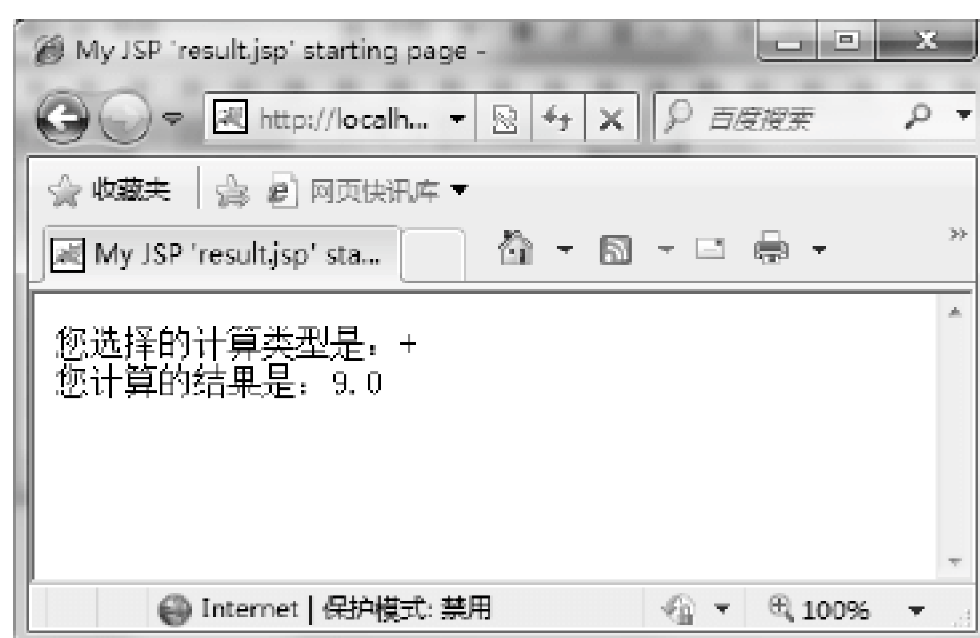


图 15.14　计算结果页

关键技术

本实例也应用了控制标签中的判断标签，相关技术参阅实例 388 关键技术部分。

设计过程

实现的代码主要是编写 index.jsp 页面中的下拉列表的代码。使用<s:select>标签实现。具体代码如下：

```
<s:form action="jisuan">
        <s:label value="简单计算器"></s:label>
        <s:textfield name="num1" label="第一个数："></s:textfield>
        <s:select name="check" list="{'+','-','*','/'}"label="计算的类型："></s:select>
        <s:textfield name="num2" label="第二个数："></s:textfield>
        <s:submit value="计算"></s:submit>
  </s:form>
```

秘笈心法

心法领悟 390：判断标签与 OGNL 语言的应用。

使用<s:if>标签与<s:else>标签判断集合 list 中的数据是否为空，其判断的过程主要是通过 OGNL 表达式中的 list.isEmpty 进行判断。

实例 391　多集合的连接

光盘位置：光盘\MR\15\391

中级

实用指数：★★☆

本实例实现的是将不同的迭代器组合在一起，使一个迭代器迭代完成后就进行下一个迭代器的迭代，直到将所有迭代器的内容迭代完成。本实例的运行结果如图 15.15 所示。

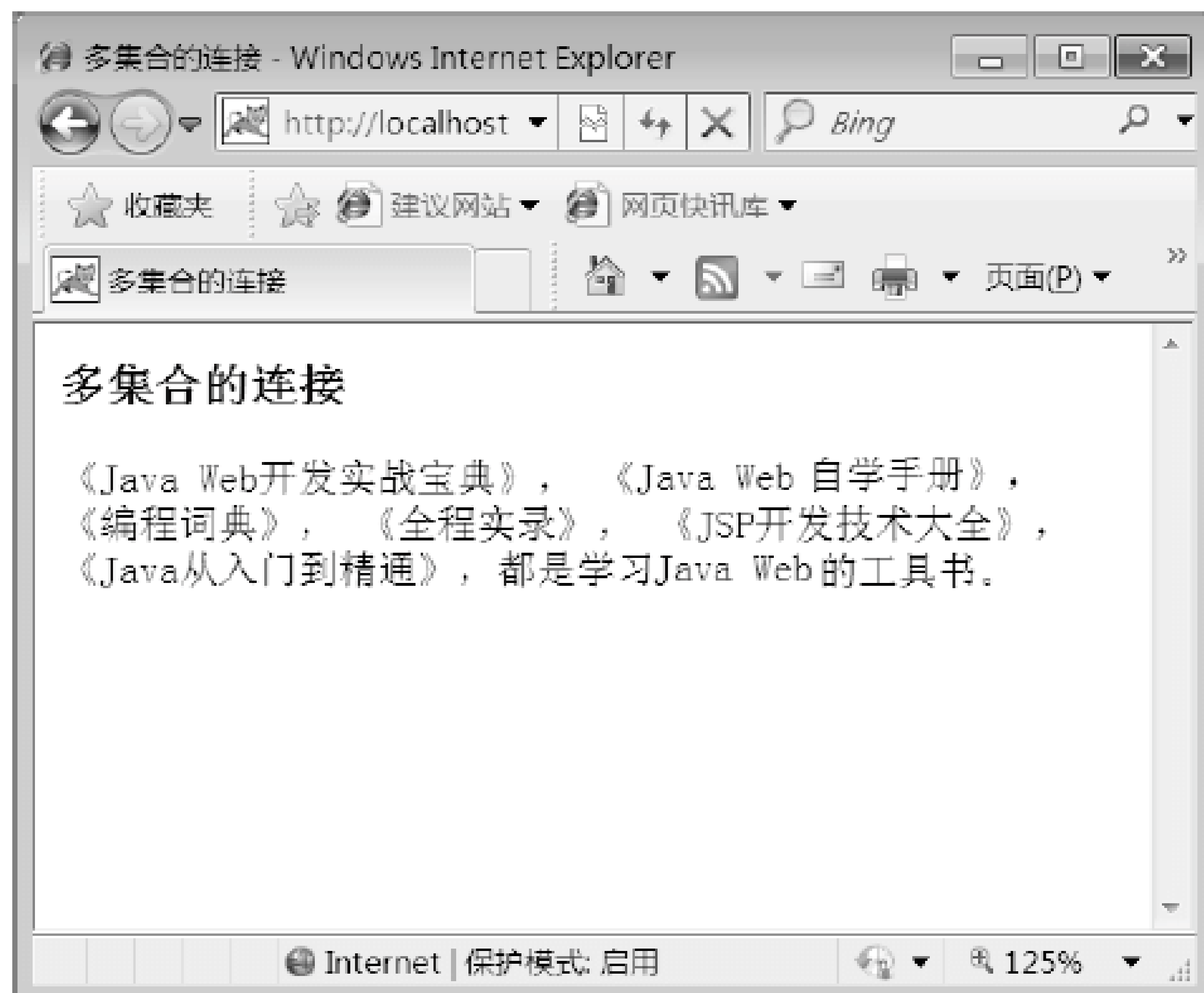


图 15.15　多集合的连接

关键技术

本实例的实现主要用到了 Struts2 标签库中的 append 标签，可通过其子标签 param 来指定要迭代的集合，而且 append 标签经常和 iterator 标签一起使用。具体语法如下：

```
<s:append id="a1">
        <s:param value="{'《Java Web 开发实战宝典》','《Java Web  自学手册》'}"/>
</s:append>
<s:iterator value="#a1" status="status">
        <s:property/>
</s:iterator>
```

设计过程

创建 index.jsp 页面，在其中引用 append 标签，在标签中添加相关子标签，进行数据的指定和输出。具体代码如下：

```
<boy>
        <h3>多集合的连接</h3>
        <s:append id="a1">
                <s:param value="{'《Java Web 开发实战宝典》','《Java Web  自学手册》'}"/>
                <s:param value="{'《编程词典》','《全程实录》'}"/>
                <s:param value="{'《JSP 开发技术大全》','《Java 从入门到精通》'}"/>
        </s:append>
        <s:iterator value="#a1" status="status">
                <!-- 判断迭代的元素是否为最后一个，如果不是，则添加逗号，如果是，则添加句号  -->
                <s:property/><s:if test="!#status.last">，</s:if><s:else>，都是学习 Java Web 的工具书。</s:else>
        </s:iterator>
</boy>
```

秘笈心法

心法领悟 391：集合连接后的顺序。

使用 append 标签实现的集合在连接后，数据的顺序是从第一个集合开始的，然后依次排列下去。集合的混合连接，后面的章节中会讲到。

实例 392　字符串的分割

光盘位置：光盘\MR\15\392

中级

实用指数：★★☆

本实例主要是实现将一个字符串按照“,”进行分割，并将分割的结果迭代出来。实例运行结果如图 15.16 所示。

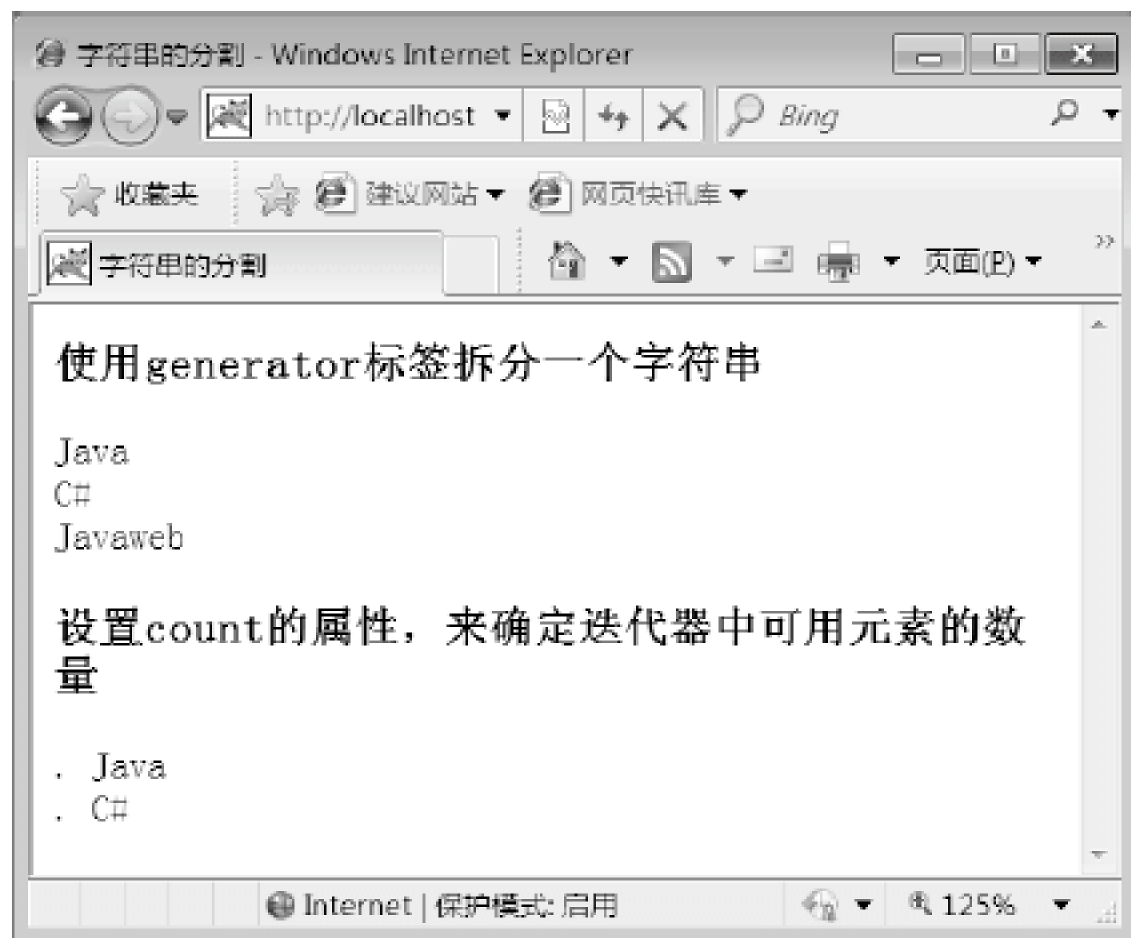


图 15.16　字符串的分割

关键技术

本实例的实现主要是应用了 Struts2 标签库中的 generator 标签。首先根据 separator 属性设定分隔符，然后将 val 属性中包含的值进行分割。接下来，可以使用 iterator 标签取出压入栈中的数据。具体语法如下：

```
< s:generator val="'Java,C#,Javaweb'" separator=",">
        <s:iterator>
                <s:property/><br>
        </s:iterator>
</s:generator>
```

设计过程

创建 index.jsp 页面，在该页面中引用 Struts2 标签库中的 generator 标签，并设定 separator 和 val 属性。具体代码如下：

```
<body>
        <h3>使用 generator 标签拆分一个字符串</h3>
<s:generator val="'Java,C#,Javaweb'" separator=",">
        <s:iterator>
                <s:property/><br>
        </s:iterator>
</s:generator>
<h3>设置 count 的属性，来确定迭代器中可用元素的数量</h3>
//由于 count 属性设置为 2，因此生成的迭代器中只有 2 个元素可用
<s:generator separator="," val="'Java,C#,Javaweb'" count="2">
        <s:iterator>
                <s:property/><br>
        </s:iterator>
</s:generator>
   </body>
```

秘笈心法

心法领悟 392：generator 标签中的 id 属性。

generator 标签中的 id 属性与其他标签中的 id 属性有些不同。如果在使用 generator 标签时使用了 id 属性，那么生成的迭代器将会保存到 pageContext 中，而不是保存到 OgnlContext 中。

实例 393	集合的混合合并 光盘位置：光盘\MR\15\393	中级 实用指数：★★☆

实例说明

本实例主要实现的是将集合中的元素混合合并到一起，然后依次输出到页面中。实例运行结果如图 15.17 所示。



图 15.17　集合的混合合并

关键技术

本实例的实现主要是应用了 Struts2 标签库中的 merge 标签。merge 标签和 append 标签在合并集合时用法类似，但不同的是在迭代时元素的顺序不同。merge 的迭代顺序是按集合中元素的顺序，依次迭代每个集合中相同顺序的元素。语法如下：

```
<s:merge id="mm">
        <s:param value="{'《Java Web 开发实战宝典》','《Java Web 范例宝典》','《模块大全》'}"/>
        <s:param value="{'《JSP 全程实录》','《JSP 开发技术大全》'}"/>
</s:merge>
        <s:iterator value="#mm" status="status">
                        <s:property/>
</s:iterator>
```

设计过程

创建 index.jsp 页面，在其中引用 Struts2 标签库中的 merge 标签，并使用迭代标签将数据迭代出来。具体代码如下：

```
<s:merge id="mm">
        <s:param value="{'《Java Web 开发实战宝典》','《Java Web 范例宝典》','《模块大全》'}"/>
        <s:param value="{'《JSP 全程实录》','《JSP 开发技术大全》'}"/>
        <s:param value="{'《Java Web 视频教学》','《Javaweb 自学手册》'}"/>
</s:merge>
<table border="1">
        <s:iterator value="#mm" status="status">
            //添加背景颜色
            <tr style="
                <s:if test="#status.odd">background-color:pink</s:if>
                <s:else>background-color:yellow</s:else>
                ">
                <td><s:property/></td>
            </tr>
        </s:iterator>
```

秘笈心法

心法领悟 393：集合混合合并的应用。

现实中有很多时候使用的都是集合的混合合并，例如，两队队列的顺次合并等，以及训练时的左前或右前一步走等。这些都可以通过编码进行模拟实现。

实例 394	筛选集合元素 光盘位置：光盘\MR\15\394	中级 实用指数：★★☆

本实例主要是实现筛选集合中的元素。在指定了所要筛选的元素的范围后，就可以将指定范围内集合中的元素输出到页面。实例运行结果如图 15.18 所示。

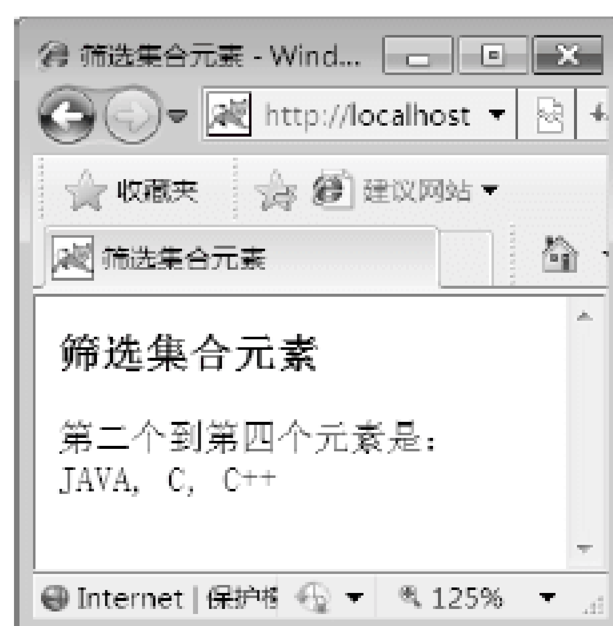


图 15.18　筛选集合元素

关键技术

实现几何元素的筛选主要是应用 Struts2 标签库中的 subset 标签。subset 标签主要是用于截取一个迭代器的子集。语法如下：

```
<s:subset source="{'PHP','JAVA','C','C++','JAVAWEB'}" start="1" count="3">
</s:subset>
```

参数说明

❶ start：设定开始元素。

❷ count：截取子集的元素数量。

设计过程

创建 index.jsp 页面，在其中编写引用 Struts2 标签库的代码；使用 subset 标签，设定 start 属性，指定开始元素；设定 count 属性，指定截取子集的大小。具体代码如下：

```
<body>
        <h3>筛选集合元素</h3>
<s:subset source="{'PHP','JAVA','C','C++','JAVAWEB'}" start="1" count="3">
                第二个到第四个元素是：<s:iterator status="status">
                        <s:property/><s:if test="!#status.last">, </s:if>
                </s:iterator>
</s:subset>
</body>
```

秘笈心法

心法领悟 394：subset 标签的 id 属性。

如果使用了 subset 标签的 id 属性，那么在截取后就会以 id 属性的值作为键值保存到 pageContext 对象中，这一点和 generator 标签的 id 属性一样。

15.3 数 据 标 签

实例 395　Action 页面的引入

光盘位置：光盘\MR\15\395

中级

实用指数：★★☆

本实例将实现在 JSP 页面中直接引用 Action，并且将 Action 对应的输出页也包含进来。实例运行结果如图 15.19 所示。

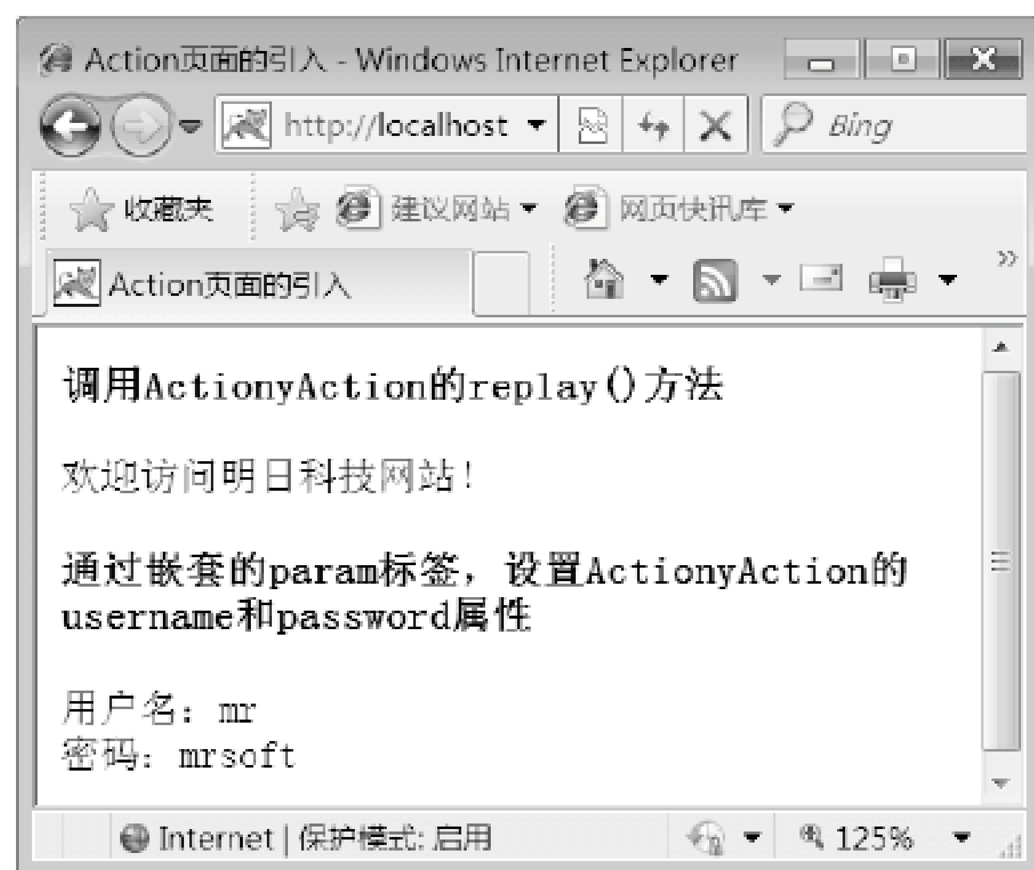


图 15.19　Action 页面的引入

关键技术

实现本实例使用的是 Struts2 数据标签中的 action 标签。action 标签允许在 JSP 页面中直接引用 action，并且可以通过设定 executeResult 属性将 action 对应的输出结果也包含到页面中。action 标签的具体语法如下：

```
<s:action name="actiony" executeResult="true" namespace="/firsts">
		<s:param name="username" value="'mr'"/>
		<s:param name="password" value="'mrsoft'"/>
</s:action>
```

主要属性说明如下。

- name：引用的 Action 名称。
- executeResult：设定是否包含 Action 的结果页。
- namespace：Action 的命名空间。

设计过程

（1）创建 ActionyAction.java 文件，在其中编写定义属性的代码和自定义方法 replay()。具体代码如下：

```
public class ActionyAction implements Action{
		private String username;
		private String password;
		//省略 get()和 set()方法
		@Override
		public String execute() throws Exception{
			return SUCCESS;
				}
		public String replay(){
			ServletActionContext.getRequest().setAttribute("result", "欢迎访问明日科技网站！ ");
			return SUCCESS;
		}
}
```

（2）创建 index.jsp 页面，在其中引入 action 标签，并配置相关属性，调用不同的 action 方法。具体代码如下：

```
<body>
   <h4>调用 ActionyAction 的 replay()方法</h4>
<s:action name="actiony!replay" executeResult="false" namespace="/firsts"/>
//获取请求对象中的属性
<s:property value="#attr.result"/>
<h4>通过嵌套的 param 标签，设置 ActionyAction 的 username 和 password 属性</h4>
<s:action name="actiony" executeResult="true" namespace="/firsts">
		<s:param name="username" value="'mr'"/>
		<s:param name="password" value="'mrsoft'"/>
</s:action>
</body>
```

秘笈心法

心法领悟 395：action 标签中的 id 属性。

在 action 标签中使用 id 属性后，action 将被放到 OgnlContext 中，在 action 标签结束后，可以通过“#id”引用 action。

实例 396	JavaBean 的引用 光盘位置：光盘\MR\15\396	中级 实用指数：★★☆

实例说明

本实例主要实现的是在页面中为 JavaBean 中的参数传值，然后获取传进去的值，在页面中输出。实例运行

结果如图 15.20 所示。

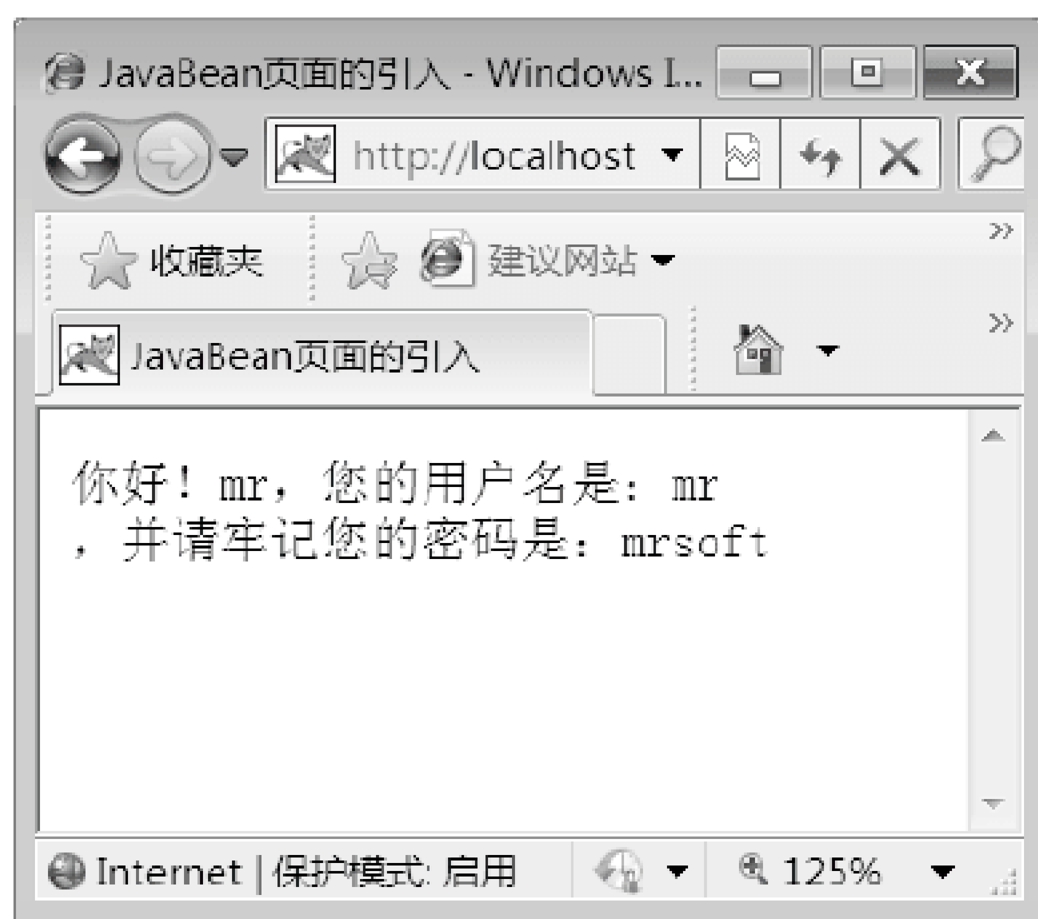


图 15.20　JavaBean 参数显示页面

关键技术

本实例的实现主要是应用了 Struts2 标签库中的 bean 标签。bean 标签用于实例化一个 JavaBean 的对象。需要注意的是，必须遵循 JavaBean 的规范。在使用 bean 标签时，标签体内可以使用多个 param 标签，用来设定 bean 的属性，但需注意要和 bean 中的 setter()方法相对应。bean 标签的语法如下：

```
<s:bean name="fe.zx.Person">
        <s:param name="username" value="'mr'" />
        <s:param name="password" value="'mrsoft'" />
        <s:property value="username"/>
        <s:property value="username"/><br>
        <s:property value="password"/>
</s:bean>
```

设计过程

（1）创建一个 JavaBean 文件，在其中定义变量和生成 get()、set()方法。具体代码如下：

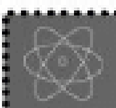
```
public class Person {
    private String username;
    private String password;
    public String getUsername() {
        return username;
    }
    //省略部分 get()和 set()方法
    }
```

（2）创建 JSP 文件，在其中引用 bean 标签，并使用相应的子标签进行参数传递。具体代码如下：

```
<body>
//应用 bean 标签
  <s:bean name="fe.zx.Person">
        //使用对应的子标签进行参数传递
        <s:param name="username" value="'mr'" />
        <s:param name="password" value="'mrsoft'" />
        你好！<s:property value="username"/>，您的用户名是：<s:property value="username"/><br>
        ，并请牢记您的密码是：<s:property value="password"/>
  </s:bean>
```

心法领悟 396：bean 标签中的 id 属性。

在 bean 标签中，id 属性无论是否指定，创建的 JavaBean 实例都会被压入到值栈中，这样在 bean 标签的内部就可以直接访问，而不用使用“#”。但是，如果指定了 id 属性，那么实例将被放到 OgnlContext 中。不过，此时在 bean 标签的外部也可以访问对象，但要使用“#”标记。



实例 397　页面日期的输出

光盘位置：光盘\MR\15\397

中级　实用指数：★★☆

实例说明

在登录网站时，经常会遇到需要注册的情况。在注册信息中，注册时间是一项很重要的内容，一般都是网页提供的系统时间。本实例将实现这一功能，运行结果如图 15.21 所示。

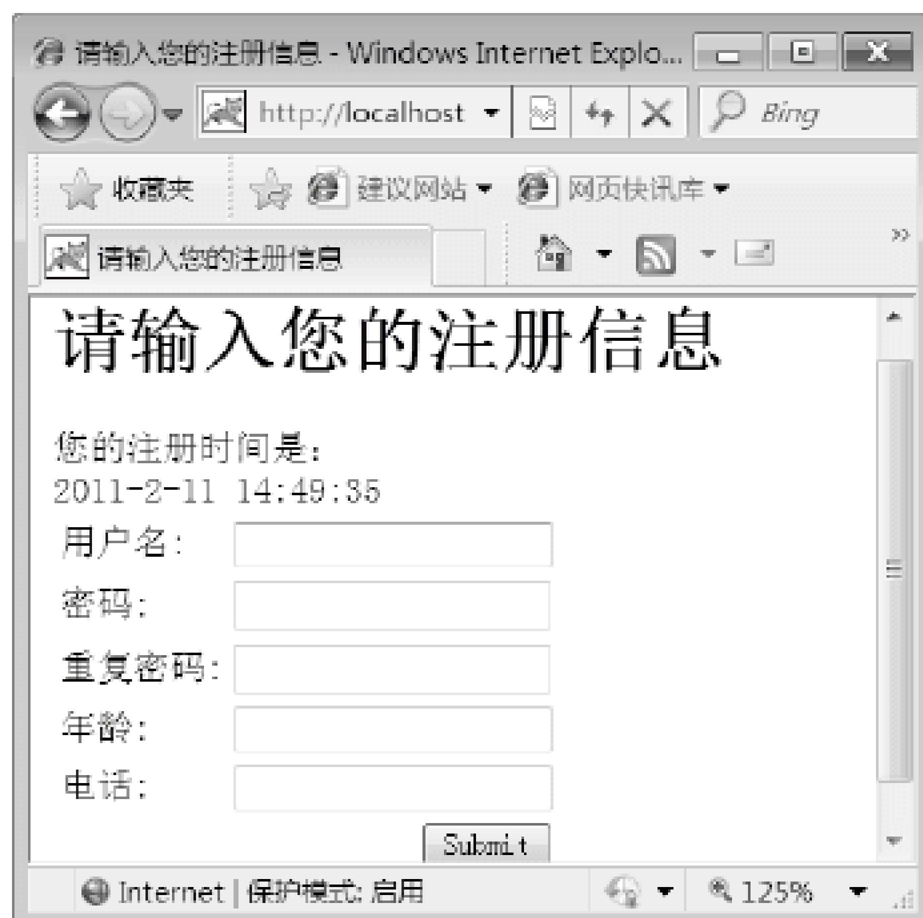


图 15.21　带时间的注册页面

关键技术

本实例主要用到了 Struts2 标签库中的 date 标签，该标签主要用于日期的一般格式化输出。语法如下：

```
<s:date name="#attr.now" />
```

date 标签的属性主要有 3 个：name 属性用于指定要格式化的日期；format 属性用于指定格式化的样式；nice 属性则是用来输出当前时间与给定日期的时间差，默认值是 false。

设计过程

创建 JSP 文件，在其中引入 date 标签，进行日期的页面输出。具体代码如下：

```
<body>
        <H1>请输入您的注册信息</H1>
<%
        //生成一个 Date 实例
        java.util.Date now = new java.util.Date();
        //将该 Date 实例设置成一个 pageContext 里的属性
        pageContext.setAttribute("now" , now);
%>
<s:form action="regist">
      <s:textfield label="用户名" name="name"/>
      <s:password label="密码" name="pass"/>
      <s:password label="重复密码" name="rpass"/>
      <s:textfield label="年龄" name="age"/>
      <s:textfield label="电话" name="phone"/>
      您的注册时间是：<br/><s:date name="#attr.now" />
      <s:submit/>
</s:form>
</body>
```

心法领悟 397：时间数据的控制。

对于一些网站中的时间数据，在开发时如果需要获取，那么就要尽量直接使用系统的时间，而不要让用户

填写。因为时间的格式在用户填写时不容易控制，容易出现错误。



实例 398　页面日期的格式化输出

光盘位置：光盘\MR\15\398

中级　实用指数：★★☆

实例说明

本实例实现了对页面输出时间的格式进行自定义，使时间的显示更符合用户的理解习惯。实例运行结果如图 15.22 所示。

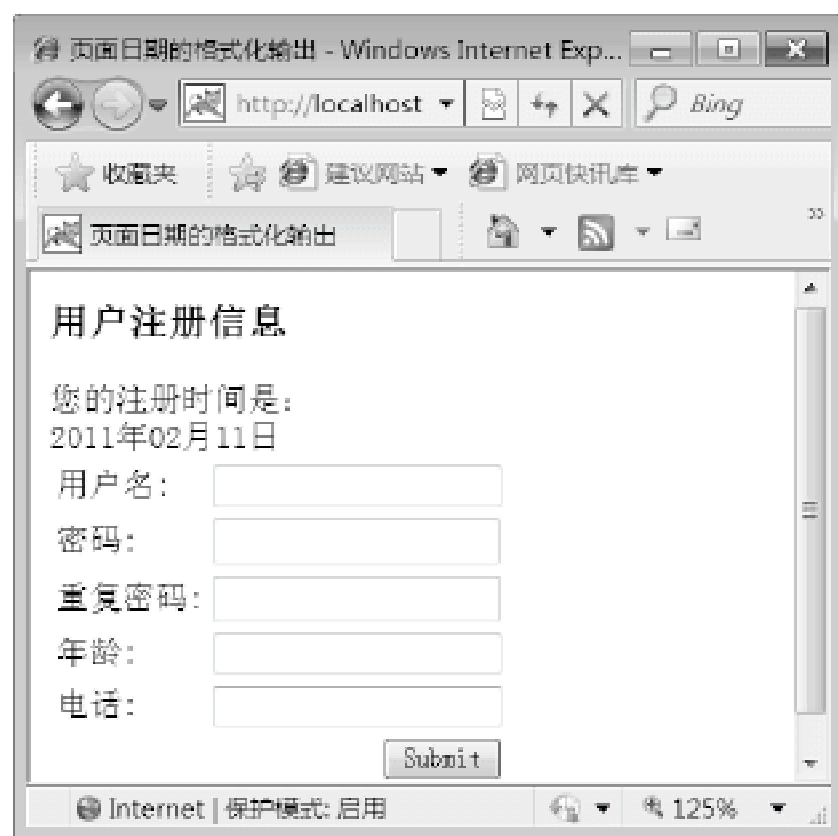


图 15.22　页面日期的格式化输出

关键技术

本实例的实现是在实例 397 的基础上进行了修改，主要应用了 Struts2 标签库中 date 标签的 format 属性。具体语法参见实例 397。

设计过程

创建一个 JSP 文件，在其中引入 date 标签，设定 date 标签的 format 属性为“yyyy 年 MM 月 dd 日”的格式。具体代码如下：

```
<body>
<%
//生成一个 Date 实例
java.util.Date now = new java.util.Date();
//将该 Date 实例设置成 pageContext 里的一个属性
pageContext.setAttribute("now" , now);
%>
 <h3>用户注册信息</h3>
 <s:form action="regist">
     <s:textfield label="用户名" name="name"/>
     <s:password label="密码" name="pass"/>
     <s:password label="重复密码" name="rpass"/>
     <s:textfield label="年龄" name="age"/>
     <s:textfield label="电话" name="phone"/>
        您的注册时间是：
            //设定 date 标签的 format 属性，实现自定义的格式
        <br/><s:date name="#attr.now" format="yyyy 年 MM 月 dd 日"/>
     <s:submit />
</s:form>
</body>
```

秘笈心法

心法领悟 398：网页中的时间格式。

不同国家的人对于时间格式的理解是不一样的。例如，在我国，人们对时间比较喜欢使用“年月日”的格式；而在英语国家中，通常使用“月日年”的格式。因此，在进行网页开发时一定要注意网站所针对的用户。



实例 399　计算日期的时间差

光盘位置：光盘\MR\15\399

中级　实用指数：★★☆

实例说明

本实例主要实现的是计算一个给定时间与现在时间的时间差。这一功能在浏览网页时会经常用到，例如，

对一些网站用户的会员等级进行确定时，就经常使用时间差。本实例的运行结果如图 15.23 和图 15.24 所示。

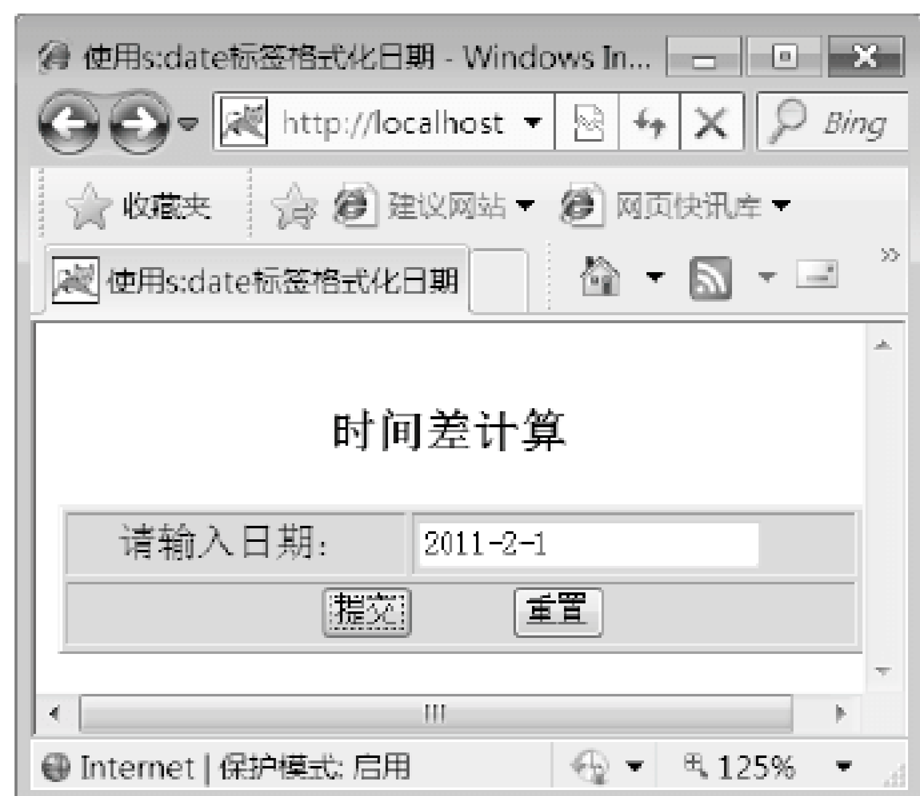


图 15.23　时间输入页面

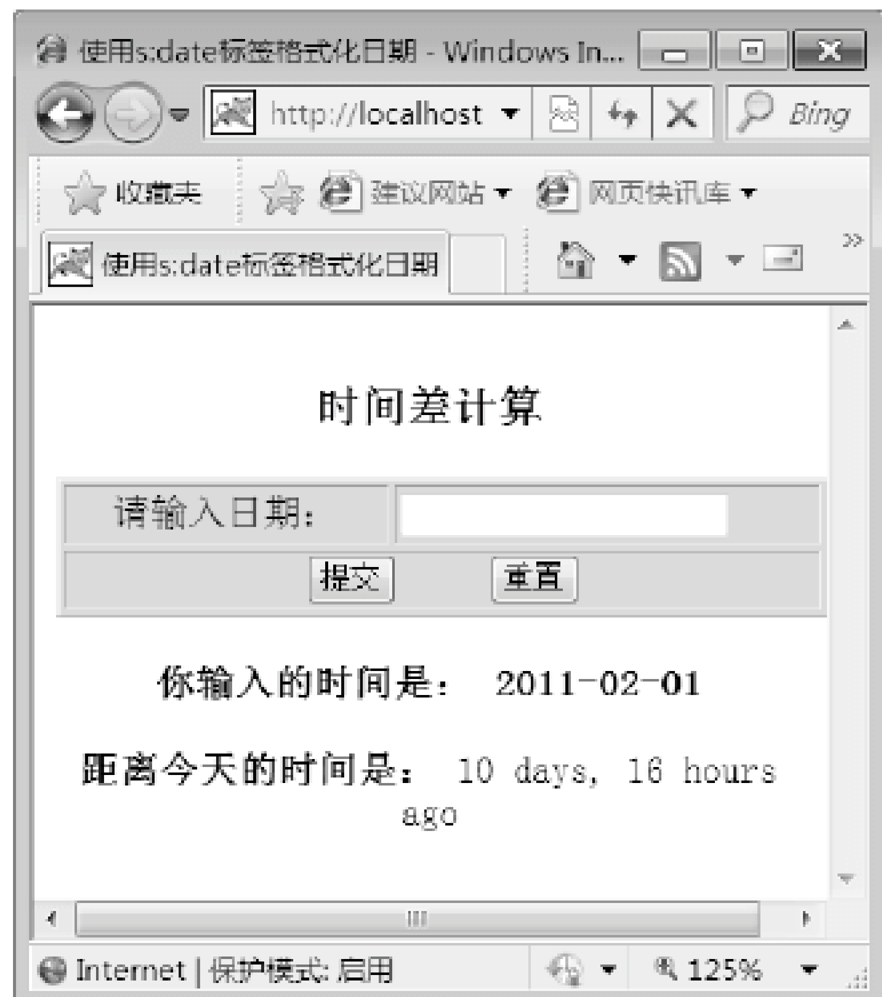


图 15.24　计算结果页面

关键技术

本实例的实现依然用到了 Struts2 标签库中的 date 标签，使用的是 nice 这一属性。nice 属性的默认值是 false，也就是不显示时间差，所以这里只要将其值改为 true 即可。

设计过程

创建一个 JSP 文件，在其中引入 date 标签，并设定 nice 属性为 true。具体代码如下：

```
<script language="javascript">
      function check()
      {
          var a=/^(\d{1,4})(-|\/)(\d{1,2})\2(\d{1,2})/;
          if(!a.test(document.form1.date.value))
           {
            alert("日期格式不正确！ ");
            return false;
           }
          else
          {
            return true;
          }
      }
      </script>
<body>
  <form action="" name="form1">
    <div align="center">
      <p><br/>
        <span class="STYLE1">时间差计算 </span></p>
      <table width="327" border="1" bgcolor="#FFCCFF">
        <tr>
          <td width="134"><div align="center">请输入日期：</div></td>
          <td width="177"><input type="text" name="date"/></td>
        </tr>
        <tr>
          <td colspan="2"><div align="center">
              <input name="submit" type="submit" onClick="return check();" value="提交"/>       //对输入的日期格式校验
              <label>
                 
              <input type="reset" name="Submit" value="重置">
              </label>
          </div></td>
        </tr>
      </table>
    </div>
  </form>
<%
```

```
    //生成一个 Date 实例
    String date=request.getParameter("date");
  if(date!=null){
    SimpleDateFormat format=new SimpleDateFormat("yyyy-MM-dd");
    //把文本框中得到的数据转换为 Date 数据类型
    Date date1=format.parse(date);
    //将该 Date 实例设置成一个 pageContext 中的属性
    pageContext.setAttribute("now", date1);
%>
<CENTER><strong>你输入的时间是：
  <s:date name="#attr.now" format="yyyy-MM-dd" nice="false" />
</strong></CENTER><br>
<CENTER><strong>距离今天的时间是：</strong>
  <s:date name="#attr.now" format="yyyy-MM-dd" nice="true" /></CENTER>     //设置 nice 属性，显示与当前日期的时间差
<%
    }
  %>
</body>
```

秘笈心法

心法领悟 399：时间数据的验证。

对于网页中要求用户必须填写的时间数据一定要进行验证。可以使用像 JavaScript 脚本语言中的正则表达式等进行验证，必要时可在页面中给出时间格式的提示。

实例 400　声明资源的国际化

光盘位置：光盘\MR\15\400

中级　实用指数：★★☆

实例说明

随着网络的国际化不断加剧，网站的开发越来越面向国际化。本实例实现的是将中文的网页国际化为英文的网页，运行结果如图 15.25 和图 15.26 所示。

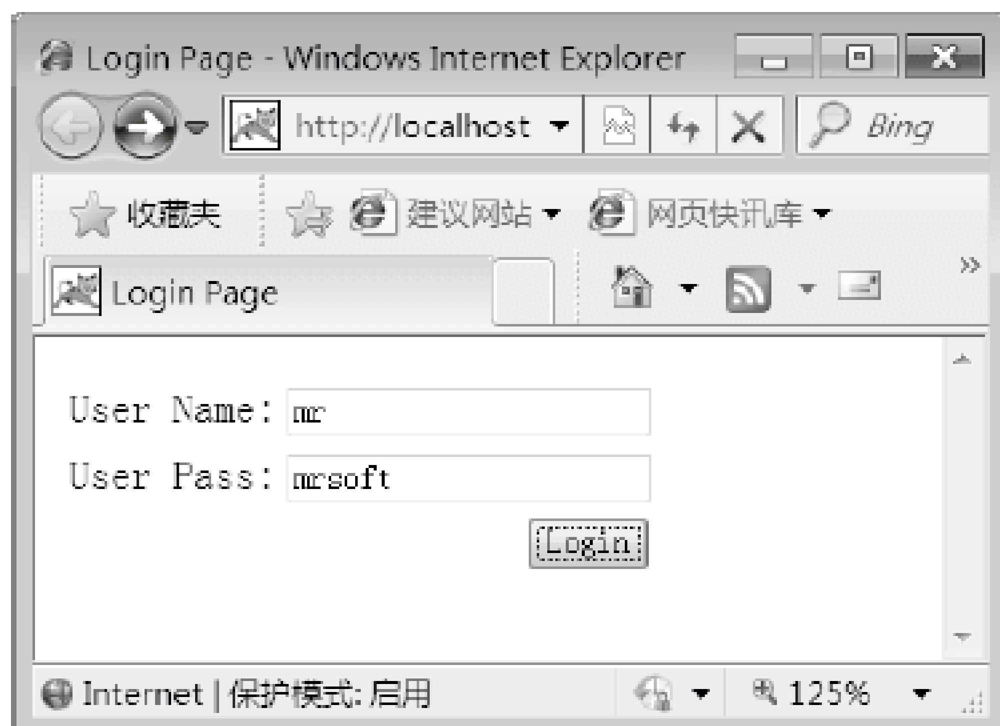


图 15.25　英文登录页面

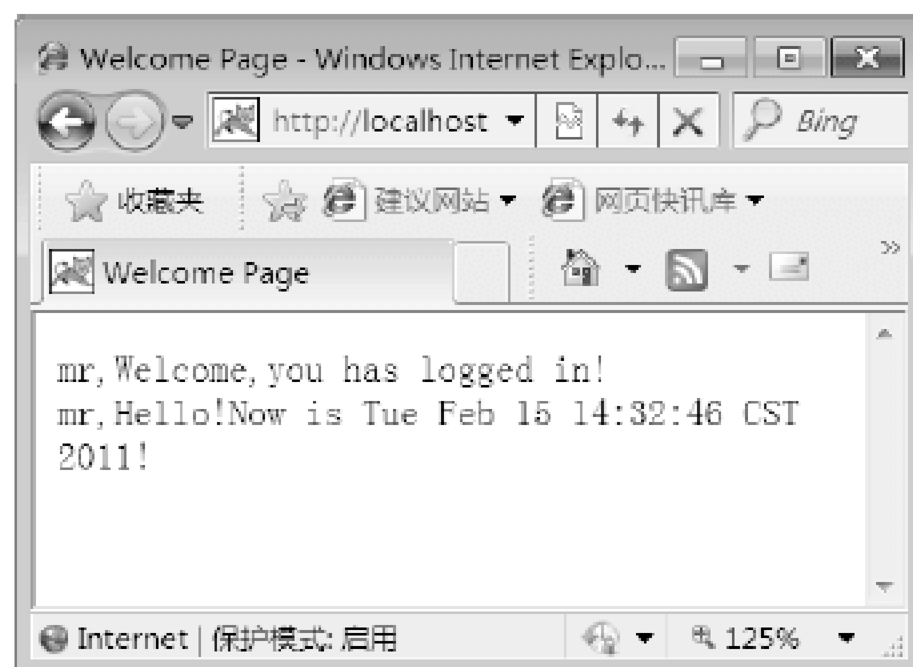


图 15.26　英文显示结果页面

关键技术

实现国际化时，可以使用 Struts2 标签库中的 i18n 和 text 标签，它们都对国际化提供了支持。本实例中主要使用了 text 标签，用于获取资源文件中的数据。具体语法如下：

```
<s:text name="welcomeMsg">
                <s:param><s:property value="username"/></s:param>
                <s:param>${d}</s:param>
</s:text>
```

设计过程

（1）创建资源文件 messageResource_en_US.properties 和其中的数据。

（2）创建 LoginAction 文件，在其中编写对于资源文件的处理方法。具体代码如下：

```
public String execute() throws Exception
    {
        ActionContext ctx = ActionContext.getContext();
    //判断用户是否可以登录
    if (getUsername().equals("mr")
            && getPassword().equals("mrsoft") )
        {
            ctx.getSession().put("user" , getUsername());
            //获取相应资源中的信息
            ctx.put("tip" , getText("succTip" , new String[]{username}));
        return SUCCESS;
    }
        else
        {
            ctx.put("tip" , getText("failTip", new String[]{username}));
        return ERROR;
    }
```

（3）创建成功登录页面，引用 text 标签。具体代码如下：

```
<body>
        ${requestScope.tip}<br/>
        <jsp:useBean id="d" class="java.util.Date" scope="page"/>
        <s:text name="welcomeMsg">
            <s:param><s:property value="username"/></s:param>
            <s:param>${d}</s:param>
        </s:text>
</body>
```

秘笈心法

心法领悟 400：程序国际化的含义。

国际化指的是在程序执行时可以根据不同的语言区以不同的语言方式进行显示。例如，操作系统是英文的，那么程序执行时就按照英语的语言习惯显示相关的信息；如果是中文的，就按照中文的语言习惯进行显示。

实例 401　JSP 页面的引入

光盘位置：光盘\MR\15\401

中级　实用指数：★★✩

本实例实现的功能是在一个页面中引入多个其他页面文件，实现页面的模块化构建，以便更好地开发和维护。本实例的运行结果如图 15.27 所示。

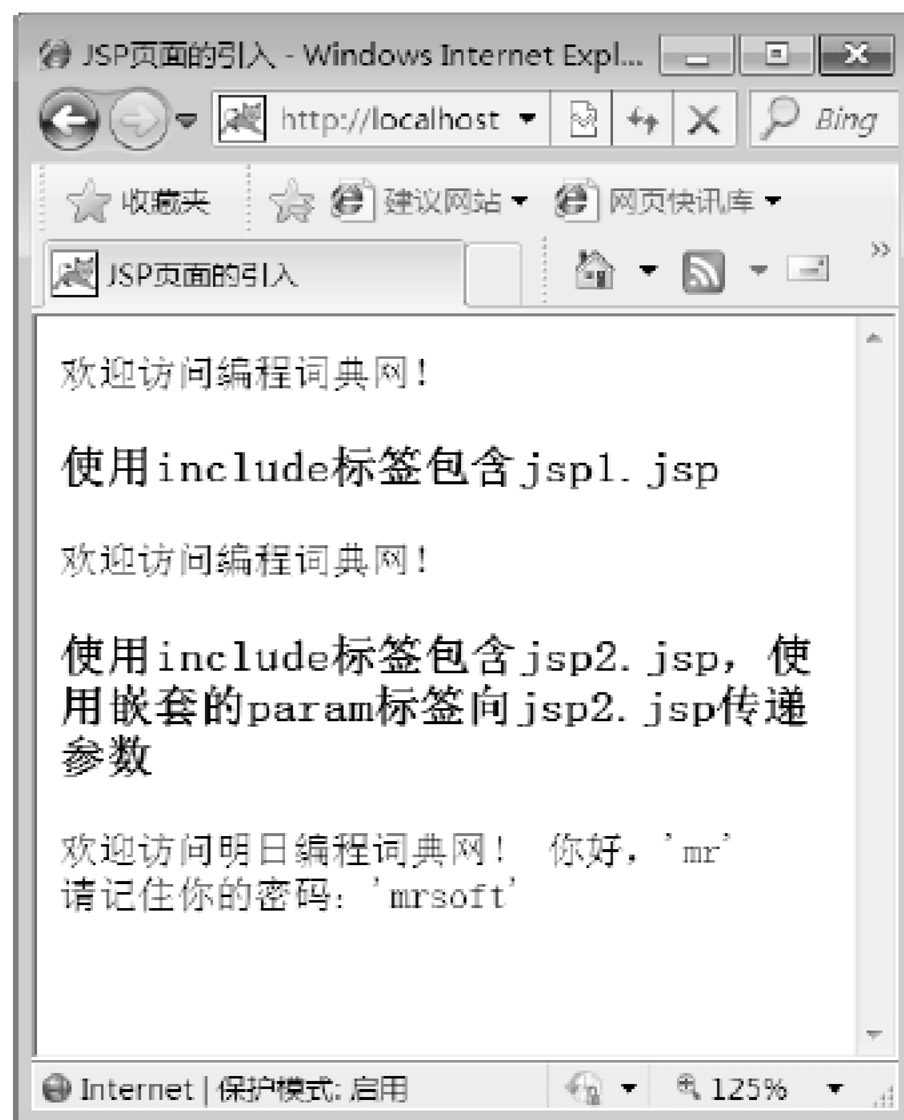


图 15.27　JSP 包含页面

关键技术

本实例的实现主要是应用了 Struts2 标签库中的 include 标签。include 标签类似于 JSP 中的<jsp:include>，用于包含一个 Servlet 或 JSP 页面；而且 include 标签体内可以包含多个 param 标签，这样就可以向被包含的页面传递请求参数。语法如下：

```
<jsp:include page="jsp2.jsp">
        <jsp:param name="username" value="'mr'"/>
        <jsp:param name="password" value="'mrsoft'"/>
</jsp:include>
```

设计过程

（1）创建两个 JSP 文件，即被引入的文件。

（2）创建一个 JSP 文件，在其中引入 include 标签，用于引入步骤（1）中创建的 JSP 文件，并且使用 param 标签进行参数传递。具体代码如下：

```
<body>
    <h3>使用 include 标签包含 jsp1.jsp</h3>
<jsp:include page="jsp1.jsp"/>
<h3>使用 include 标签包含 jsp2.jsp，使用嵌套的 param 标签向 jsp2.jsp 传递参数</h3>
<jsp:include page="jsp2.jsp">
        <jsp:param name="username" value="'mr'"/>
        <jsp:param name="password" value="'mrsoft'"/>
</jsp:include>
  </body>
```

秘笈心法

心法领悟 401：网页的模块化开发。

在网站开发中，如果遇到比较复杂的网页，需要编写大量的代码，可以将其分隔开，然后分块开发，最终进行组合。这样不仅开发时不易产生混乱，而且在以后的维护中也会更加方便。

实例 402	页面间数据的传递 光盘位置：光盘\MR\15\402	中级 实用指数：★★☆

实例说明

本实例实现的是在页面间传递 Java 代码，并且在页面中输出。实例运行结果如图 15.28 所示。

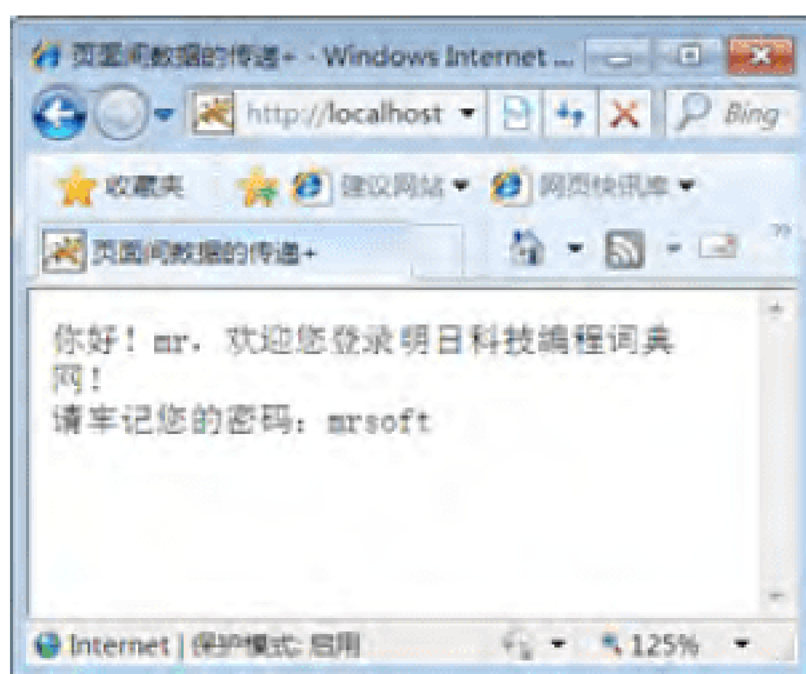


图 15.28　页面间数据的传递

关键技术

本实例的实现用到了常用的 param 标签。该标签通常被用作其他标签的子标签，用于为其他标签提供参数。其主要属性有两个，一个是 name，用于确定要设置的参数名称；另一个是 value，用来设置要设置的参数值。具体语法如下：

```
<s:param name="username" value="'mr'" />
<s:param name="password" value="'mrsoft'"/>
```

设计过程

创建一个 JSP 文件，并在其中引入 param 标签，设置其 name 和 value 属性。具体代码如下：

```
<body>
  <s:bean name="fe.zx.PersonBean" >
              <s:param name="username" value="'mr'" />
              <s:param name="password" value="'mrsoft'"/>
              你好！<s:property value="username"/>，欢迎您登录明日科技编程词典网！<br />
              请牢记您的密码：<s:property value="password" />
</s:bean>
  </body>
```

秘笈心法

心法领悟 402：对于固定参数的设定。

有时网页中一些固定的参数值是不需要用户输入的，那么开发时就可以在网页中通过一些网页的传参标签等进行传入，而且在网页中可以获取输出，这样可以很好地提高开发效率。

实例 403　页面数据的设定　中级

光盘位置：光盘\MR\15\403　实用指数：★★★

实例说明

本实例实现的是为指定的不同范围内的变量进行赋值，并且将所赋的值在页面中进行输出。实例运行结果如图 15.29 所示。

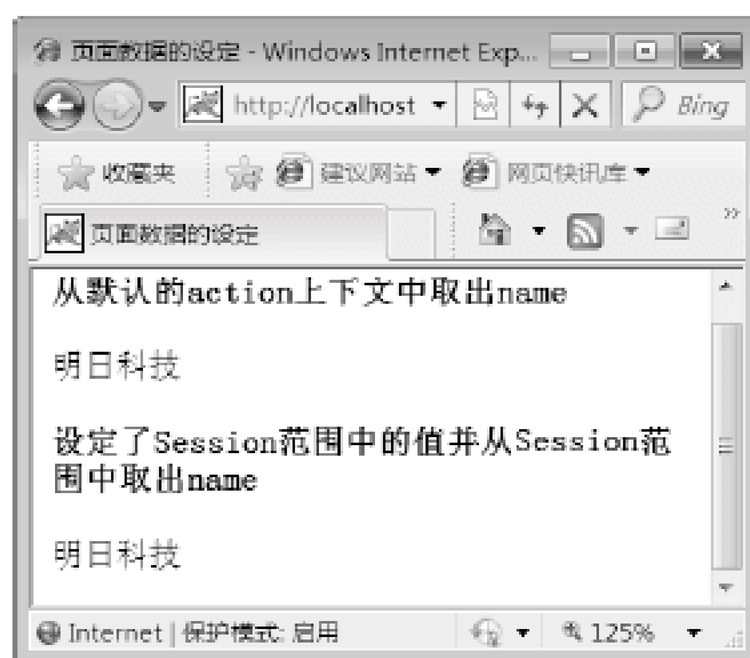


图 15.29　页面数据的设定

关键技术

本实例的实现主要应用了 set 标签。set 标签在某些情况下是比较实用的，尤其在页面中多次引用同一个复杂的表达式时，使用 set 标签可以将这个表达式赋给一个变量，然后直接引用该变量。

在 set 标签中最主要的就是 scope 属性，用来指定值的范围，主要取值有 page、request、session 和 application。set 标签的具体语法如下：

```
<s:set name="name" value="user.username"/>
//将表达式 user.username 的值保存到 Session 范围中
<s:set name="name" value="user.username" scope="session"/>
```

（1）创建一个 Action 文件，在其中设定需要的数据和处理代码。

（2）创建 JSP 文件，在其中引用 set 标签，并且对 scope 属性设定不同的属性值。具体代码如下：

```
<body>
       //将表达式 user.username 的值保存到默认范围中，即 action 范围
       <s:set name="name" value="user.username"/>
```

```
        <h4>从默认的 action 上下文中取出 name</h4>
        <s:property value="#name"/>
        //将表达式 user.username 的值保存到 Session 范围中
        <s:set name="name" value="user.username" scope="session"/>
        <h4>设定了 Session 范围中的值并从 Session 范围中取出 name</h4>
        <s:property value="#session.name"/>
    </body>
```

秘笈心法

心法领悟 403：使用 set 标签的好处。

- 可以大幅提高代码的性能，因为表达式只需要计算一次。
- 还可以使代码具有很高的可读性。

实例 404 变量值的页面输出

光盘位置：光盘\MR\15\404

中级

实用指数：★★☆

实例说明

在网站开发中，经常需要将数据输出到页面中。本实例实现的是将指定的数值输出到页面中，如图 15.30 所示。

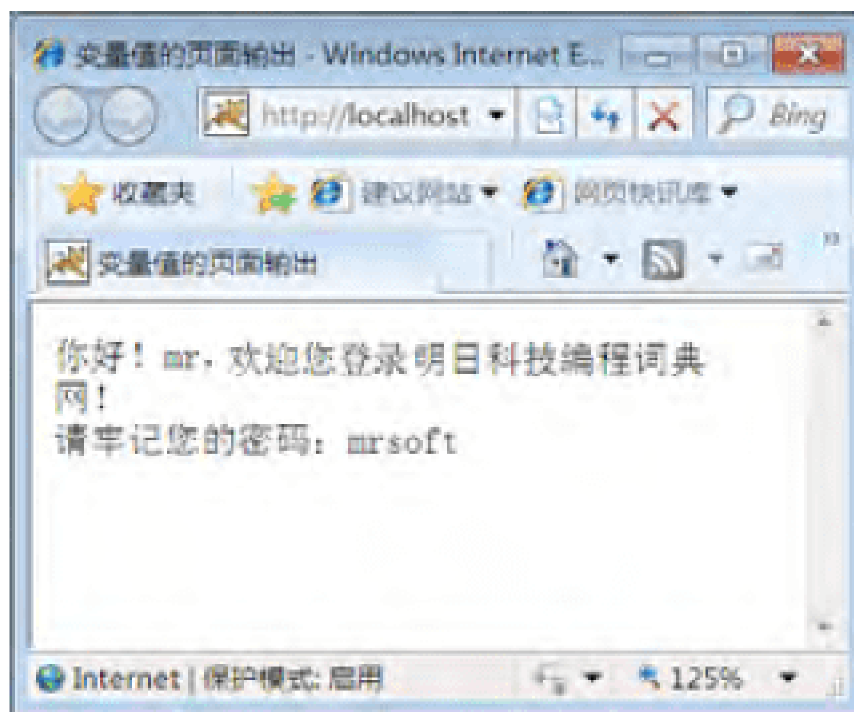


图 15.30 变量输出页面

关键技术

本实例中使用的 property 标签在其他实例中已经大量使用过了，下面具体介绍一下。property 标签的作用是输出指定的值，指定值需要通过设定标签的 value 属性来实现。property 标签的具体使用语法如下：

```
<s:property value="username"/>
//输出 password 的值
<s:property value="password" />
```

设计过程

（1）创建一个 Java 文件，在其中定义变量并且编写 get()和 set()方法。

（2）创建 JSP 文件，在其中引用 property 标签，设定 value 属性。具体代码如下：

```
<body>
  <s:bean name="fe.zx.PersonBean" >
        <s:param name="username" value="'mr'" />
        <s:param name="password" value="'mrsoft'"/>
        你好！<s:property value="username"/>，欢迎您登录明日科技编程词典网！<br />
        请牢记您的密码：<s:property value="password" />
    </s:bean>
  </body>
```


心法领悟 404：property 标签的取值。

property 标签如果没有指定 value 属性，将直接取出栈顶值进行输出；但如果指定了栈顶值，就会输出指定的值。

15.4 表 单 标 签

实例 405　**表单的输出**　**中级**

光盘位置：光盘\MR\15\405　　实用指数：★★★

实例说明

本实例中实现的是一个简单的用户注册页面，用户在填写了自己的注册信息后，单击“注册”按钮，即可在另一页面中显示用户的注册信息，如图 15.31 和图 15.32 所示。

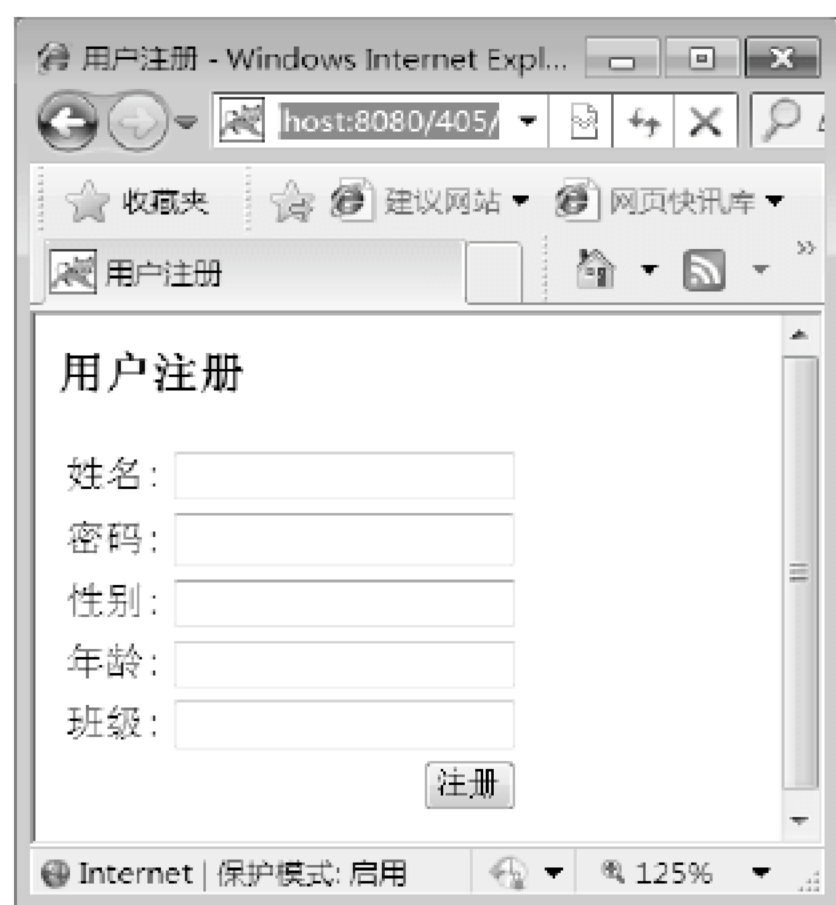


图 15.31　注册页面

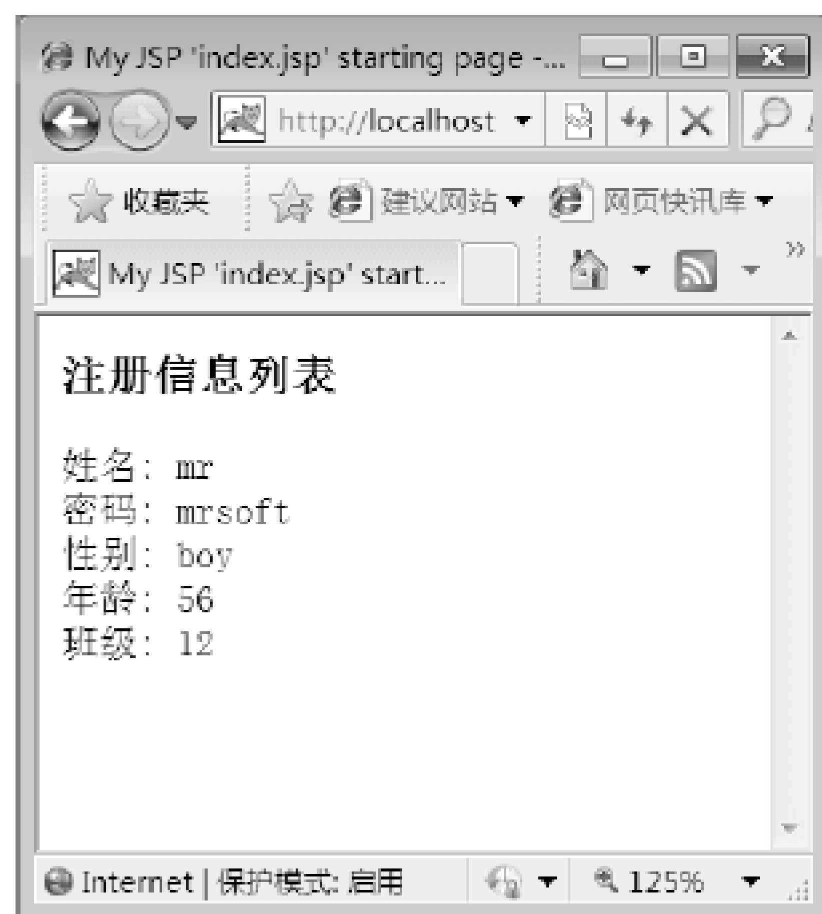


图 15.32　信息显示页面

关键技术

本实例中主要应用的是 Struts2 标签库中的表单标签<s:form>，form 标签中的 action 属性用于说明本页面的提交数据将由哪一个 Action 处理。form 标签的使用语法如下：

```
<s:form action="regist">
        <s:textfield name="username" label="姓名"></s:textfield>
        //子标签
        <s:submit value="注册"></s:submit>
  </s:form>
```

（1）创建 index.jsp 页面，引用 form 标签。具体代码如下：

```
<body>
<h3>用户注册</h3>
    <s:form action="regist">
        <s:textfield name="username" label="姓名"></s:textfield>
          <s:password name="password" label="密码"></s:password>
          <s:textfield name="sex" label="性别"></s:textfield>
          <s:textfield name="age" label="年龄"></s:textfield>
          <s:textfield name="grade" label="班级"></s:textfield>
          <s:submit value="注册"></s:submit>
     </s:form>
</body>
```

（2）创建 RegisterAction.java 文件，在其中定义相关属性和方法。具体代码如下：

```
public class RegisterAction extends ActionSupport{
    private String username;
    private String password;
```

```
private String sex;
private String age;
private String grade;
   //省略 get()和 set()方法
public String execute()
{
          return "success";
}
   }
```

秘笈心法

心法领悟 405：页面的多表单。

在开发中进行页面设置时，不一定一个页面就是一个表单，可以在一个页面中实现多个表单，而且可以实现多表单对不同或相同 Action 的提交。

实例 406　用户名的填写　中级

光盘位置：光盘\MR\15\406　实用指数：★★★

实例说明

本实例实现的是对用户填写的用户名进行输出，以便用户更好地确认是自己在登录，这对于用户的安全登录是一种很重要的保护措施。本实例的运行结果如图 15.33 所示。

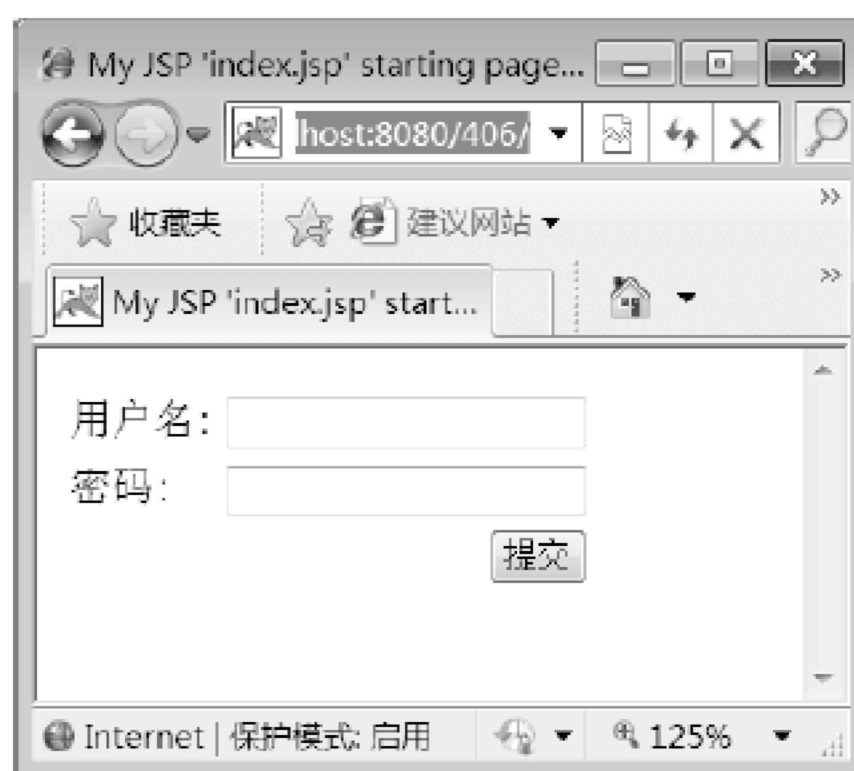


图 15.33　用户名的填写

关键技术

本实例主要是应用 Struts2 标签库中的 textfield 标签来实现页面的文本框，在文本框中实现用户名的填写。textfield 标签的使用语法如下：

```
<s:textfield name="username" label="用户名"></s:textfield>
```

该标签的主要属性是 name，在参数传递时可以作为参数的键值标识。

创建 index.jsp 页面，在其中引入 textfield 标签，实现文本框。具体代码如下：

```
<body>
    <s:form action="lo">
        <s:textfield name="username" label="用户名"></s:textfield>
          <s:textfield name="password" label="密码"></s:textfield>
          <s:submit value="提交"></s:submit>
    </s:form>
    <s:if test="username!=null">
                    你输入的用户名是：<s:property value="username" />
    </s:if>
    </body>
```

秘笈心法

心法领悟 406：xhtml 标题中标签输出的特殊列。

在 xhtml 标题中，textfield 标签会输出一个有两列的表（其实不止 textfield 标签，其他的标签也一样），一列是 label 属性的值，另一列是表单元素。

实例 407　简单的用户登录页面

光盘位置：光盘\MR\15\407

中级

实用指数：★★☆

实例说明

本实例主要是实现一个简单的登录页面，用户只需填写简单的用户名和密码即可进行登录，如图 15.34 所示。

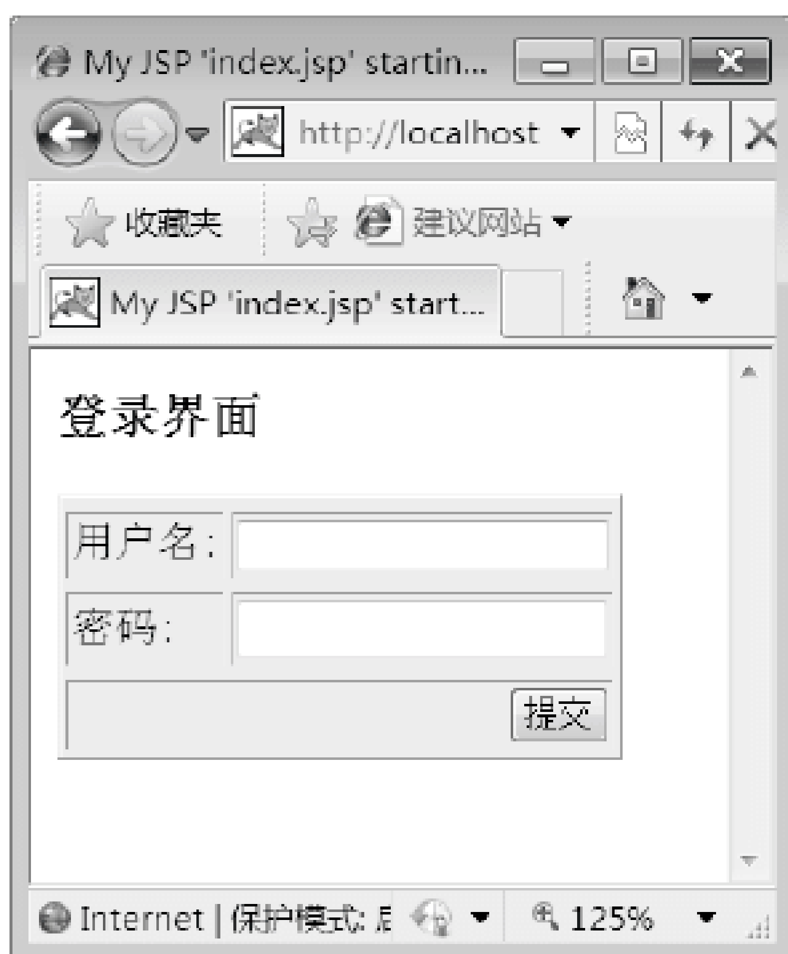


图 15.34　简单的用户登录页面

关键技术

本实例是在实例 406 的基础上增加了密码框这一功能。实现密码框使用的是 password 标签。语法如下：

```
<tr> <s:password name="password" label="密码"></s:password></tr>
```

password 标签除了公共属性外，还有其他几个主要属性——maxlength 属性，可以设定输入密码的最大长度；size 属性，用于设定可以看到的密码位数；showPassword 属性，可以设定是否显示密码。

设计过程

创建 index.jsp 页面，在其中引入 password 标签，实现密码框。关键代码如下：

```
<body>
    <h3>登录界面</h3>
     <s:form>
     <table border="1" bgcolor="yellow">
         <tr ><s:textfield name="username" label="用户名"></s:textfield></tr>
          <tr> <s:password name="password" label="密码"></s:password></tr>
            <tr><s:submit value="提交"></s:submit></tr>
            </table>
     </s:form>
</body>
```

秘笈心法

心法领悟 407：密码的验证。

密码对于网站的安全是十分重要的，在对密码进行验证时，可以与数据库中的数据进行对比，重要的密码可以在代码中进行先期的验证。

实例 408 本地文件的浏览

光盘位置：光盘\MR\15\408

中级

实用指数：★★☆

实例说明

在上传文件时常常要浏览本地文件，然后选择需要上传的文件。本实例将实现对本地文件的浏览和选择，用户单击“浏览”按钮后，就会进入本地文件的选择界面。实例运行结果如图 15.35 所示。

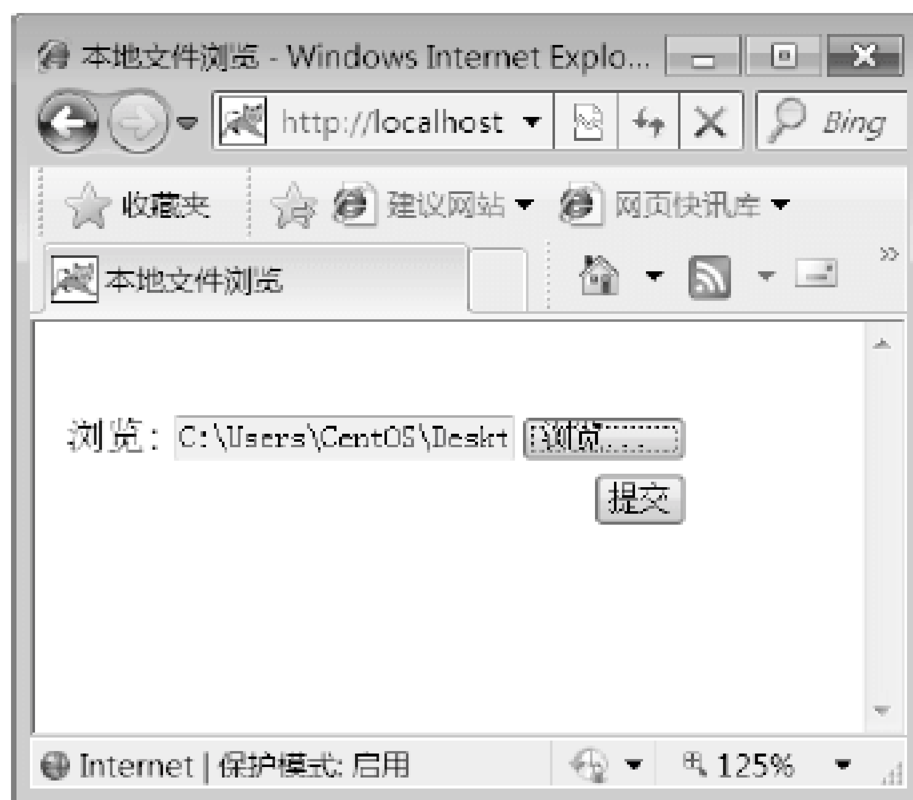


图 15.35　本地文件浏览

关键技术

实现本实例主要应用的是 Struts2 标签库中的 file 标签。file 标签的使用很简单。语法如下：

```
<s:file name="file" label="浏览"></s:file><br/>
```

file 标签中的 accept 属性很重要，该属性主要用于指定可接收文件的 MIME 类型。

设计过程

创建 index.jsp 文件，并在其中写入 file 标签实现本地文件的浏览功能。具体代码如下：

```
<body>
    <s:form>
      <s:file name="file" label="浏览"></s:file><br/>
          <s:submit value="提交"></s:submit>
    </s:form>
</body>
```

秘笈心法

心法领悟 408：Struts2 中上传文件的相关设置。

Struts2 中会默认使用 javax.servlet.context.tempdir 作为工作目录，上传的文件会被临时保存在这个目录中，上传结束后会自动删除，所以 action 中会将文件复制出来。在开发时，可以通过覆盖 struts.multipart.saveDir 属性来修改工作目录。在此要注意的是，Struts2 默认的上传文件最大是 2097152 字节，如果文件过大，则会提示错误。

实例 409 数据的默认选择

光盘位置：光盘\MR\15\409

中级

实用指数：★★☆

本实例实现的功能很简单，就是在用户进行数据提交时，如果对数据的准确性没有要求，那么可以使用网站给出的默认数据值，如图 15.36 所示。



图 15.36　数据的默认选择结果

关键技术

本实例的实现技术很简单，就是应用了标签中的 value 属性，直接在 value 属性中填写属性值，如果用户不改变就是默认的数据。

设计过程

创建 index.jsp 文件，在其中引入相关 Struts2 标签，并设定 value 属性值。具体代码如下：

```
<body>
<h3>数据的默认选择</h3>
    <s:form>
        <s:textfield name="username" label="姓名" value="张三"></s:textfield>
          <s:textfield name="sex" label="性别" value="男"></s:textfield>
          <s:textfield name="age" label="年龄" value="22"></s:textfield>
          <s:textfield name="grade" label="班级" value="20805 班"></s:textfield>
          <s:submit value="注册"></s:submit>
  </s:form>
  </body>
```

秘笈心法

心法领悟 409：网站数据的选择。

对于网站默认数据的选择，一般会根据不同的地域文化和网站的属性进行选择。默认数据本身就是为了节省用户的操作和减少不良数的产生，所以在开发时必须注意默认数据的选择。

实例 410　页面中单选按钮的实现

光盘位置：光盘\MR\15\410

中级

实用指数：★★☆

实例说明

本实例主要是实现页面中的单选按钮，供用户对一些唯一性的数据进行选择。实例运行结果如图 15.37 所示。



图 15.37　单选按钮的实现效果

关键技术

本实例的实现主要是应用 Struts2 框架中的 radio 标签，该标签的工作方式与后文将要讲到的 select 标签类似。radio 标签的具体语法如下：

```
<s:radio name="sex" label="性别" list="#{0:'男',1:'女'}"/>
```

其中，list 属性指定的是可以用来选择的值。

设计过程

创建 index.jsp 页面，在其中引入 Struts2 的 radio 标签，设定 list 属性的值。具体代码如下：

```
<body>
<h3>用户注册</h3>
    <s:form >
        <s:textfield name="username" label="姓名"></s:textfield>
        <s:password name="password" label="密码"></s:password>
        <s:radio name="sex" label="性别"   list="#{0:'男',1:'女'}"/>
        <s:textfield name="age" label="年龄"></s:textfield>
        <s:textfield name="grade" label="班级"></s:textfield>
        <s:submit value="注册"></s:submit>
    </s:form>
</body>
```

秘笈心法

心法领悟 410：默认值的使用。

通常对一些值的选择在开发时是可以给出默认选择的，例如性别，一般都是默认选择男。在设定时，只要在 radio 标签中加入 value 属性即可。语法如下：

```
<s:radio name="sex" label="性别" value="1"    list="#{0:'男',1:'女'}"/>
```

radio 标签会将 value 属性的值与 list 属性指定的 Map 值进行比较。上面代码中 value 值为 1，那么就会默认匹配键值为 1 的值。

实例 411 实现表单的提交

光盘位置：光盘\MR\15\411

中级

实用指数：★★☆

实例说明

本实例将创建一个用户注册的表单，在单击“注册”按钮后可以实现表单数据的提交。实例运行结果如图 15.38 所示。



图 15.38　提交表单

本实例主要是应用 form 标签中的 action 属性和 submit 标签来实现。通过 action 属性中指定的 action 名称确定处理的 action，使用 submit 标签实现提交按钮。具体语法如下：

```
<body>
<h3>用户注册</h3>
```

```
    <s:form action="regist">
        //省略表单内容
            <s:submit value="注册"></s:submit>
</s:form>
</body>
```

设计过程

（1）创建用于处理表单提交内容的 Action，名称为 RegisterAction。具体代码如下：

```
public class RegisterAction extends ActionSupport{
    private String username;
    private String password;
    private String sex;
    private String age;
    private String grade;
      //省略 get()和 set()方法
      //action 默认执行方法
    public String execute()
    {
      if(sex.equals("0")){
            sex="男";
      }
      else{
            sex="女";
      }
                  return "success";
    }
}
```

（2）创建表单页面，引用<s:submit>标签，实现提交按钮。具体代码如下：

```
<body>
<h3>用户注册</h3>
    <s:form action="regist">
        //省略表单内容
            <s:submit value="注册"></s:submit>
    </s:form>
</body>
```

秘笈心法

心法领悟 411：提交表单数据。

对于表单的提交，主要是对其中的数据进行提交。在提交数据时，对数据的识别一般都是用键值来标识，所以数据标识键值的唯一性一定要注意，否则容易产生对数据提交和处理的不准确性，这一点应该引起开发者的注意。

实例 412	实现下拉列表框 光盘位置：光盘\MR\15\412	中级 实用指数：★★☆

实例说明

本实例实现的是下拉列表框。运行程序，即可将“地区”以下拉列表框的形式显示出来，本实例的运行结果如图 15.39 所示。

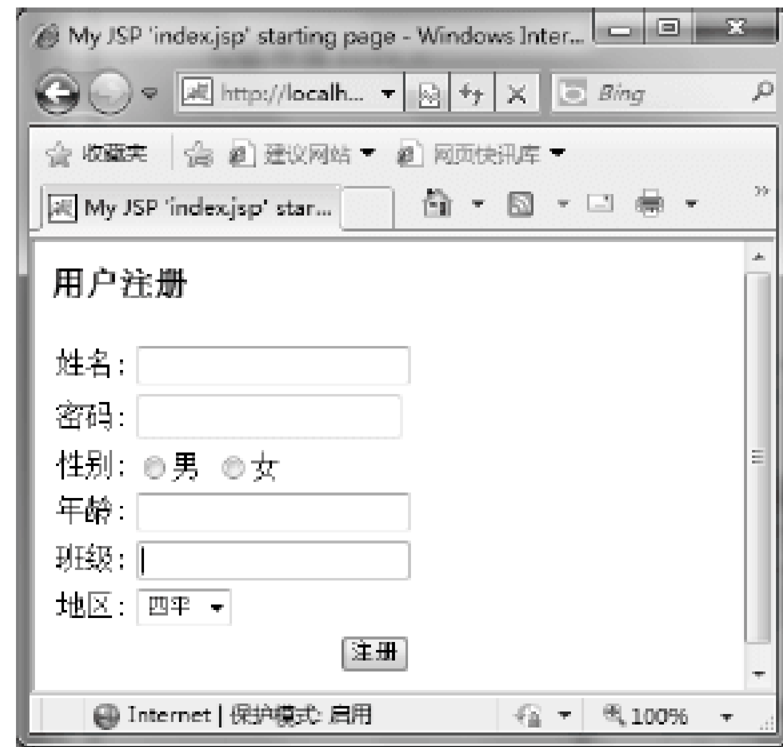


图 15.39　实现下拉列表框

关键技术

要实现下拉列表框，就要用到 Struts2 标签库中的 select 标签。该标签的常用属性如下。

- list：要迭代的集合。
- listKey：指定使用集合中对象的哪一个属性作为选项的 value。
- listValue：指定使用集合中对象的哪一个属性作为选项的内容。
- multiple：是否创建一个多选列表，默认为 false。

- size：设置下拉列表框中可显示的选项个数。

设计过程

创建用于显示下拉列表框的 index.jsp 页面。具体代码如下：

```
<s:form>
        <s:textfield name="username" label="姓名"></s:textfield>
        <s:password name="password" label="密码"></s:password>
        <s:radio name="sex" label="性别" list="#{0:'男',1:'女'}"/>
        <s:textfield name="age" label="年龄"></s:textfield>
        <s:textfield name="grade" label="班级"></s:textfield>
        <s:select name="city" list="{'四平','松原','九台','白城','延边'}" label="地区"></s:select>          //创建下拉列表框
        <s:submit value="注册"></s:submit>
</s:form>
```

##

心法领悟 412：list 属性值的类型。

list 属性包括 Collection、Map、Enumeration、Iterator、Array 类型。如果 list 属性值是一个 Map，则 Map 的键会变为选项的 value，Map 的值会变为选项的内容。

实例 413　具有自动完成功能的下拉列表框

光盘位置：光盘\MR\15\413

中级

实用指数：★★☆

实例说明

本实例实现的是具有自动完成功能的下拉列表框。运行程序，当在下拉列表框中输入内容，如“河”字时，下拉列表框能够根据所输入内容自动进行筛选并将相关数据显示出来。本实例的运行结果如图 15.40 所示。

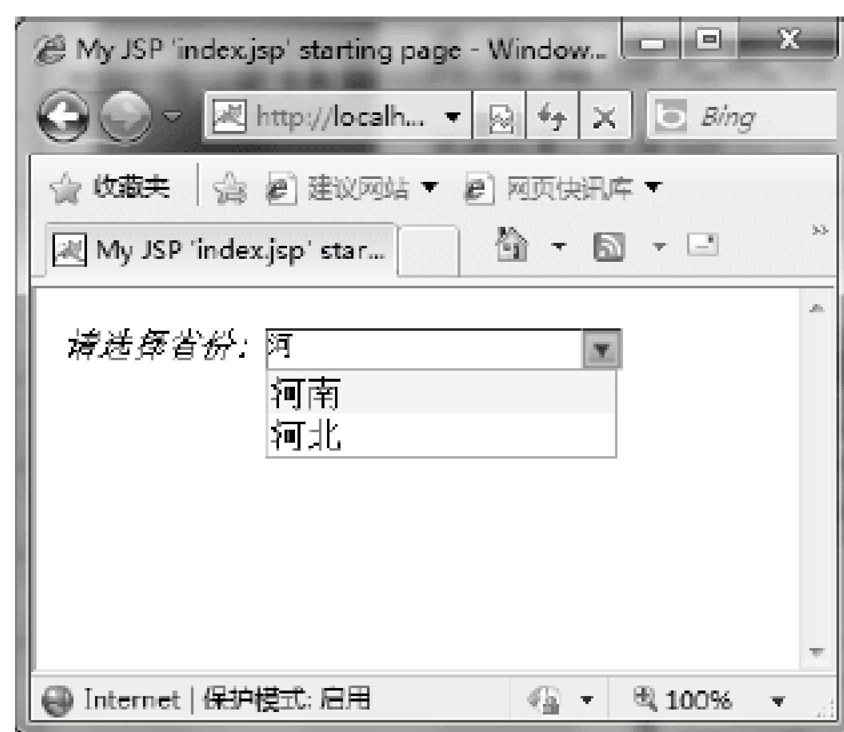


图 15.40　具有自动完成功能的下拉列表框

关键技术

要实现具有自动完成功能的下拉列表框，就要用到 Struts2 标签库中的 autocomplete 标签，该标签能够根据所输入的内容自动筛选下拉列表框中的相关数据。

设计过程

创建实现自动筛选功能的 index.jsp 页面。具体代码如下：

```
<%
    List city=new ArrayList();                                        //创建一个 list，并为该 list 添加数据
    city.add("河南");
    city.add("河北");
    city.add("山东");
    city.add("山西");
    city.add("湖南");
    city.add("湖北");
    city.add("广东");
    city.add("广西");
    city.add("贵州");
    city.add("云南");
    request.setAttribute("city",city);
%>
  <s:form action="">
    <sx:autocompleter name="city" label="请选择省份" list="%{#request.city}"></sx:autocompleter>   //获取 list 中数据，并自动筛选
  </s:form>
```

秘笈心法

心法领悟 413：autocomplete 标签的使用。

使用 autocomplete 标签必须使用 Ajax 主题，因为该标签用到了 Dojo 库。

实例 414　使用动态数据的下拉列表框

光盘位置：光盘\MR\15\414

中级

实用指数：★★☆

实例说明

本实例实现的是使用动态数据的下拉列表框。运行程序，当在下拉列表框中输入内容，如“湖”字时，下拉列表框能够根据所输入内容动态地将相关数据显示出来。本实例的运行结果如图 15.41 所示。

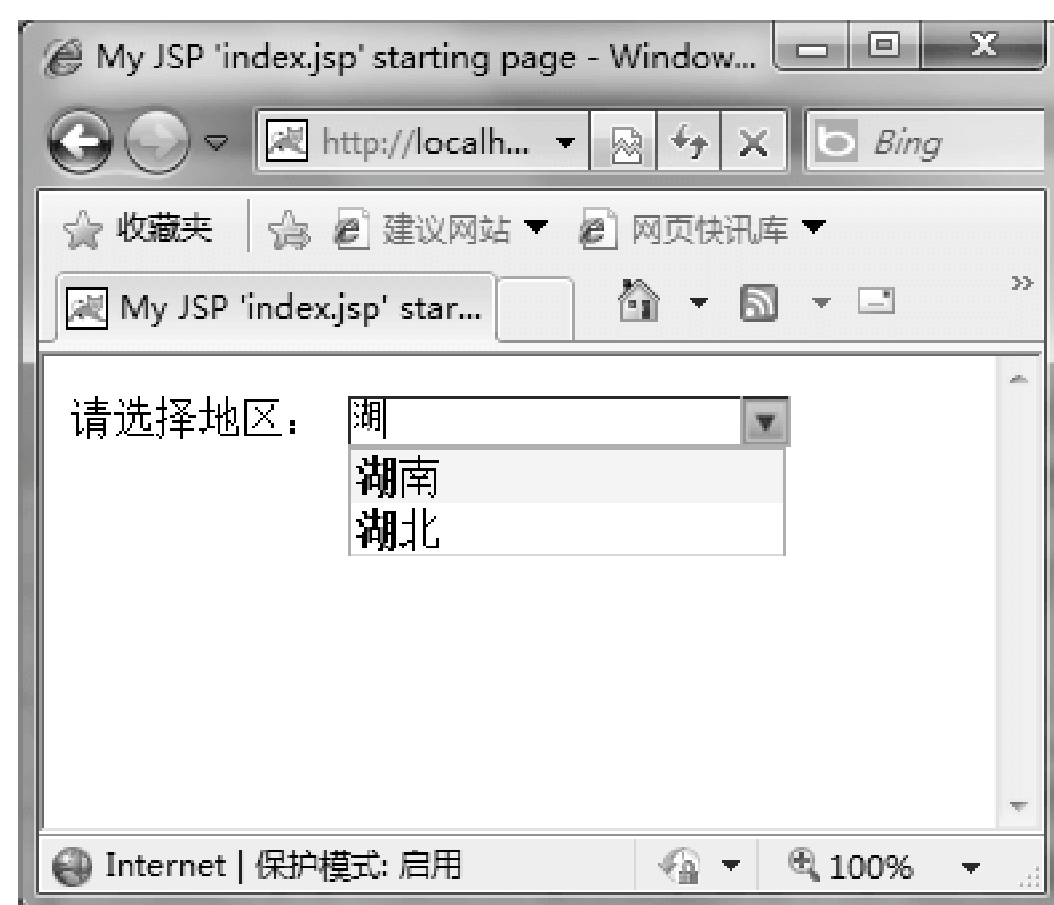


图 15.41　使用动态数据的下拉列表框

关键技术

要实现使用动态数据的下拉列表框，就要用到 Struts2 标签库中的 autocomplete 标签。该标签通过 href 属性指定获取数据的 URL，该 URL 会根据当前输入的内容动态地筛选内容，返回下拉列表框数据。

设计过程

（1）创建用于存储数据的 data.jsp 页面。具体代码如下：

```
<%
    out.clear();
      request.setCharacterEncoding("UTF-8");                                  //设置编码格式
      response.setHeader("Pragma", "no-cache");
      response.setHeader("Cache-Control", "no-cache");
      response.setDateHeader("Expires", 0);                                   //禁止缓存
      String[] citys = {"河南","河北","湖南","湖北","广东","广西","山东","山西","福州","福建"};   //创建数据数组
      String city = request.getParameter("city");
    if (city == null)
            city = "";
      StringBuffer buffer = new StringBuffer();
      for (int i = 0; i < citys.length; i++) {                                //遍历所有地区
            if (citys[i].startsWith(city)) {                                  //判断是否包含输入的字符串
                  if (buffer.length() != 0)
                 buffer.append(",");
                  buffer.append("['" + citys[i] + "']");
            }
      }
      Thread.sleep(500);
      out.print("[" + buffer + "]");                                          //显示数据
%>
```

（2）创建用于获取数据的 index.jsp 页面，具体代码如下：

```
<body>
<s:url id="dataUrl" value="/data.jsp" />                                    //URL 请求
请选择地区：
<sx:autocompleter name="city"   href="%{dataUrl}"                           //指定 href 属性
loadOnTextChange="true" loadMinimumCount="1" autoComplete="false"
showDownArrow="true"/>
</body>
```

秘笈心法

心法领悟 414：通过 autocomplete 标签加载动态数据。

一般可以使用该标签的 list 属性获取静态的数据，但是当下拉列表框中的数据很庞大时，只能使用动态数据，然后通过 autocomplete 标签进行加载。

实例 415 复选框的实现

光盘位置：光盘\MR\15\415

中级

实用指数：★★☆

实例说明

本实例实现的是复选框。运行程序，即可看到个人爱好以复选框的形式显示出来。本实例的运行结果如图 15.42 所示。

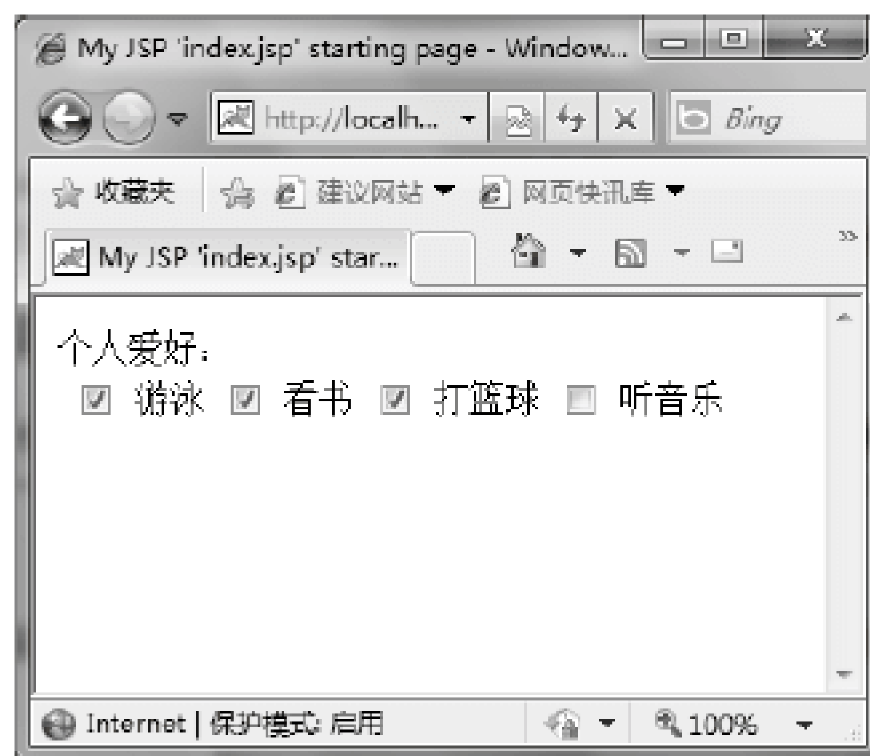


图 15.42 页面复选框

关键技术

要实现复选框，要用到 Struts2 标签库中的 checkboxlist 标签。该标签的主要属性如下。

- list：要迭代的集合。
- listKey：指定使用集合中对象的哪一个属性作为选项的值。
- listValue：指定使用集合中对象的哪一个属性作为选项的内容。

设计过程

创建用于显示复选框的 index.jsp 页面。具体代码如下：

```
<body>
  <s:form action="">
          个人爱好：<s:checkboxlist name="like" list="{'游泳','看书','打篮球','听音乐'}"></s:checkboxlist>     //多选框的显示
  </s:form>
  </body>
```

秘笈心法

心法领悟 415：checkboxlist 标签中的 list 属性值。

其中的每个 list 属性值都要用单引号（''）括起来，同时属性值之间用“,”隔开。

实例 416 实现可填写的复合框

光盘位置：光盘\MR\15\416

中级

实用指数：★★☆

本实例实现的是可填写的复合框。运行程序，即可看到个人爱好以复合框的形式显示出来（其中包括一个文本框和一个下拉列表框）。本实例的运行结果如图 15.43 所示。

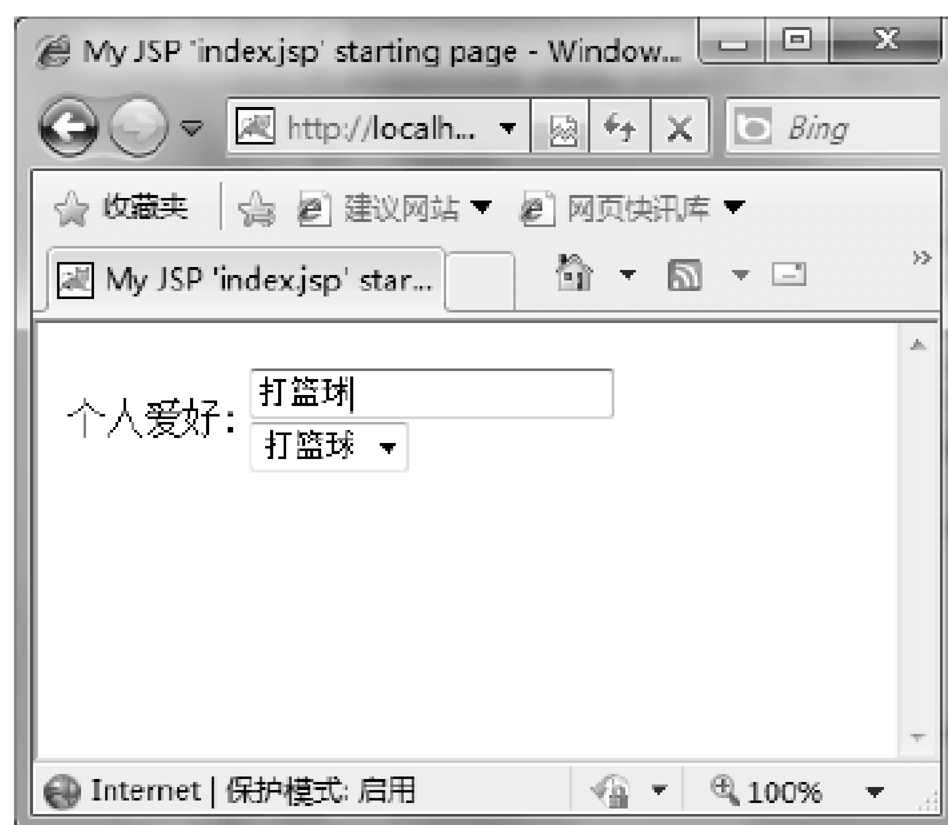


图 15.43　可填写的复合框

关键技术

要实现复合框，就要用到 Struts2 标签库中的 combobox 标签。combobox 标签兼具选择与编辑的功能，既可以选择下拉列表框中的值，也可以填写下拉列表框中没有的值。

设计过程

创建用于显示可填写的复合框的 index.jsp 页面。具体代码如下：

```
<body>
  <s:form action="">
    <s:combobox name="like" list="{'游泳','看书','打篮球','听音乐'}" label="个人爱好"></s:combobox>        //复合框的显示
  </s:form>
</body>
```

秘笈心法

心法领悟 416：checkboxlist 标签与 Map 集合的结合。

使用 Map 集合作为数据源时，可以使用 key 和 value 值分别代表 Map 对象的 Key 和 Value 作为复选框的 value。

实例 417	日期选择器 光盘位置：光盘\MR\15\417	中级 实用指数：★★☆

本实例实现的是日期选择器。运行程序，单击右侧的按钮，可以显示一个日历。本实例的运行结果如图 15.44 所示。



图 15.44　日期选择器

关键技术

要实现日期选择器，要用到 Struts2 标签库中的 datetimepicker 标签。该标签是专门输入日期时间的输入框，它自带了一个日历，同时还可以指定日期显示格式。

设计过程

创建用于显示日期选择器的 index.jsp 页面。具体代码如下：

```
<body>
    <s:form action="">
      <sx:datetimepicker name="date" displayFormat="yyyy/MM/dd" label="请选择日期" />          //显示日期选择器同时指定显示格式
    </s:form>
</body>
```

秘笈心法

心法领悟 417：日期格式的设置。

日期格式可以通过属性 displayFormat 来设置。

实例 418 联动选择框

光盘位置：光盘\MR\15\418

中级

实用指数：★★☆

实例说明

本实例实现的是联动选择框。运行程序，可以看到页面中有两个下拉列表框，在上面的下拉列表框中选择一个省份，则下方下拉列表框中对应的市会相应地变动。本实例的运行结果如图 15.45 所示。

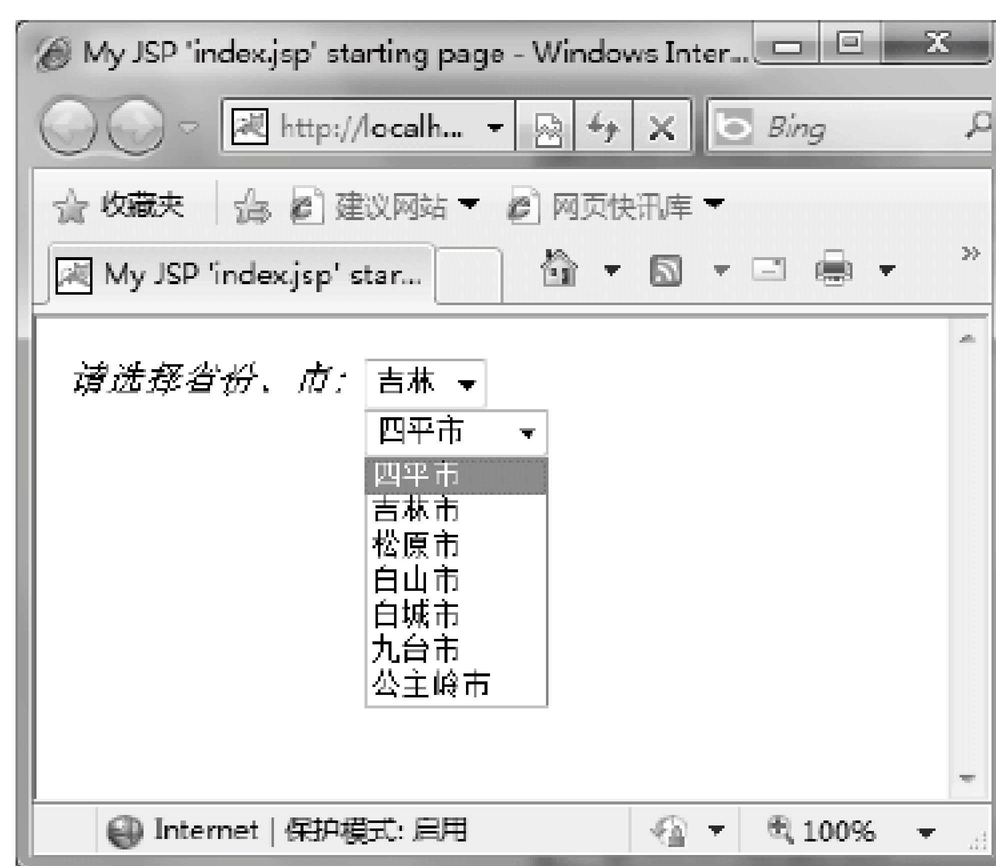


图 15.45　联动选择框

关键技术

要实现联动选择框，要用到 Struts2 标签库中的 doubleselect 标签。此标签用于输出关联的两个 HTML 列表框，第二个列表框显示的内容随第一个列表框选中的选项而变化。其常用属性如下。

- list：要迭代的集合。
- doubleList：该属性对 list 属性中的每一个元素求值，返回一个迭代集合。

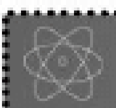
设计过程

创建用于显示联动选择框的 index.jsp 页面。具体代码如下：

```
<body>
    <s:form action="k">
    <s:doubleselect name="province" list="{'河南','吉林'}"                    //创建联动选择框
      doubleName="city"  doubleList="top=='河南'?{'郑州市','新乡市','商丘市','许昌市','周口市','安阳市','开封市'}:{'四平市','吉林市','松原市','白山市','白城市','九台市','公主岭市'}" label="请选择省份、市"/>
      </s:form>
</body>
```

心法领悟 418：doubleselect 标签的 doubleName 属性值。

在本实例中为了演示效果，直接将 doubleName 的属性值设置为 city，而在实际应用中，该属性值是与 Action 的属性相对应的。



实例 419　多级数据选择框

光盘位置：光盘\MR\15\419

中级

实用指数：★★☆

实例说明

本实例实现的是多级数据选择框。运行程序，可以看到页面中左、右各有一个列表框，既可以把“已经选中的好友”中的姓名放到“人员列表”中，也可以把“人员列表”中的姓名放到“已经选中的好友”中。本实例的运行结果如图 15.46 所示。

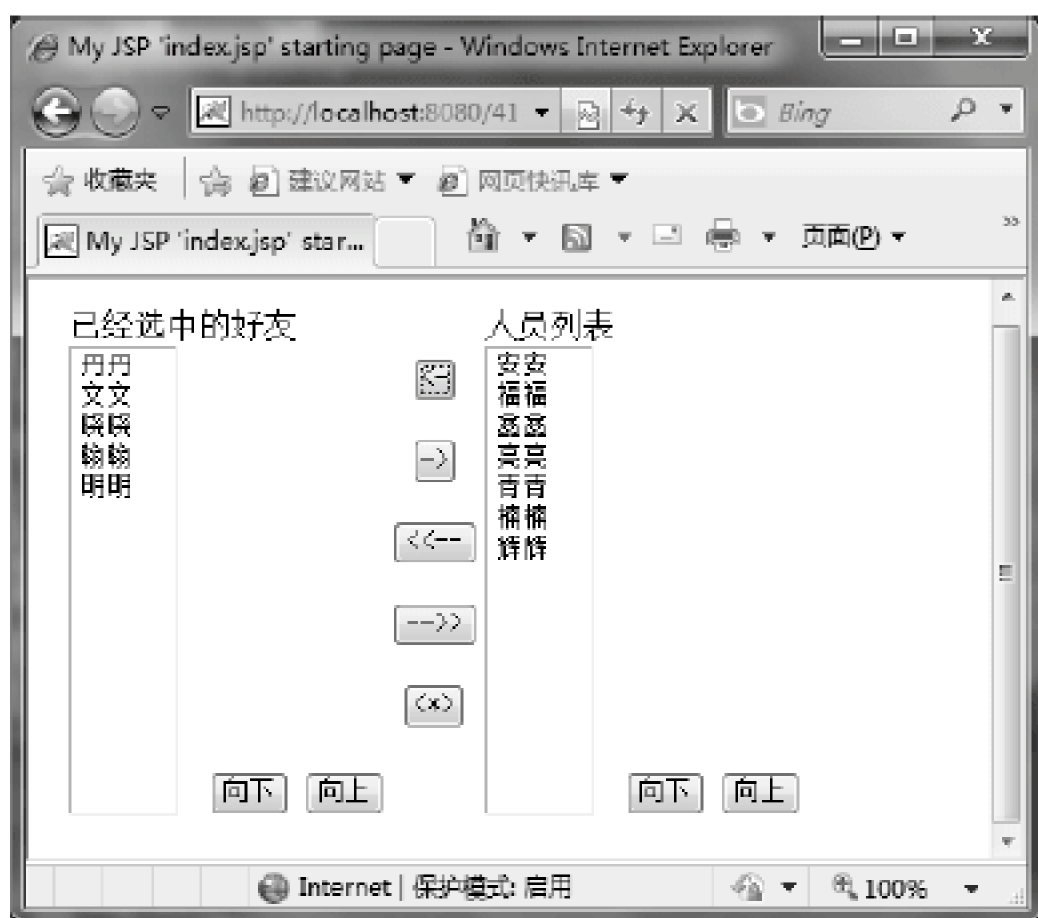


图 15.46　多级数据选择框

关键技术

要实现多级数据选择框，要用到 Struts2 标签库中的 optiontransferselect 标签。其常用属性如下。

- list：要迭代的集合，使用集合中的元素设置各个选项。
- doubleList：要迭代的集合，使用集合中的元素设置各个选项。
- leftUpLabel：设置左边列表框中向上移动按钮的文本。
- leftDownLabel：设置左边列表框中向下移动按钮的文本。
- rightUpLabel：设置右边列表框中向上移动按钮的文本。
- rightDownLabel：设置右边列表框中向下移动按钮的文本。
- leftTitle：设置左边列表框的标题。
- rightTitle：设置右边列表框的标题。

设计过程

创建用于显示多级数据选择框的 index.jsp 页面。具体代码如下：

```
<body>
    <s:form>
      <s:optiontransferselect name="friends" doubleList="{'安安','福福','鑫鑫','明明','亮亮','青青','楠楠','辉辉'}"
            list="{'丹丹','文文','晓晓','翰翰'}" doubleName="person" leftUpLabel="向上"
            leftDownLabel="向下" rightDownLabel="向下" rightUpLabel="向上"
            leftTitle="已经选中的好友" rightTitle="人员列表" />
    </s:form>
  </body>
```

心法领悟 419：optiontransferselect 标签的组成。

该标签是由两个 select 标签和可以移动标签中内容的按钮组成，当表单进行提交时，将会提交两个列表框中选中的内容。

第16章

Hibernate 框架基础

- ⏭ 操作实体对象
- ⏭ HQL 与 QBC 检索方式

16.1　操作实体对象

实例 420	将实体对象保存到数据库 光盘位置：光盘\MR\16\420	中级 实用指数：★★★

实例说明

Hibernate 提供了强大、高性能的对象到关系型数据库的持久化服务，可以在不写 SQL 语句的情况下将客户端输入的数据保存到数据库中。运行本实例，在如图 16.1 所示页面中的“留言人”文本框中输入留言人，在“留言内容”文本框中输入留言内容，单击“提交”按钮，即可将留言信息保存到数据库中。



图 16.1　将留言信息保存到数据库

关键技术

Session 是 Hibernate 持久化操作的基础，提供了众多持久化的方法，如 save()方法、update()方法和 delete()方法等，通过这些方法可以完成对象的添加、修改和删除等持久化操作。在本实例中，主要应用 save()方法实现添加操作。save()方法的语法如下：

```
session.save(Object)
```

- session：指的是 Session 实例，可以通过以下代码创建。

```
Configuration config = new Configuration().configure();
SessionFactory sessionFactory = config.buildSessionFactory();
Session session = sessionFactory.openSession();
```

- Object：用于指定持久化类的对象，即指定将哪个持久化对象保存到数据库中。

（1）创建保存留言信息的数据表 tb_message，表结构如图 16.2 所示。

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	INT(10)	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
writer	VARCHAR(45)	✓		☐ BINARY	NULL	留言人
content	VARCHAR(200)	✓		☐ BINARY	NULL	留言内容
sendTime	TIMESTAMP	✓			CURRENT_TIMESTAMP	留言时间

图 16.2　tb_message 的数据表结构

（2）在 MyEclipse 中创建项目并导入 Hibernate 包。

（3）创建 Hibernate 配置文件 hibernate.cfg.xml，该文件中存放着数据库连接驱动程序类、登录数据库的用户名/密码、映射实体类配置文件的位置等，Hibernate 初始化时会自动在 classes 路径中寻找该文件，并读取其中的配置信息，为后期数据库操作做准备。代码如下：

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
          "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
          "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="dialect">                              //指定数据库使用的方言
            org.hibernate.dialect.MySQLDialect
        </property>
        <property name="connection.url">                       //指定连接数据库的 URL 地址
            jdbc:mysql://localhost:3306/db_database16
        </property>
        <property name="connection.username">root</property>   //指定连接数据库的用户名
        <property name="connection.password">111</property>    //指定连接数据库的密码
        <property name="connection.driver_class">              //指定连接数据库的驱动
            com.mysql.jdbc.Driver
        </property>
        <mapping resource="com/wgh/model/TbMessage.hbm.xml" /> //指定持久化类映射文件
    </session-factory>
</hibernate-configuration>
```

说明：在 hibernate.cfg.xml 文件的头部，即<?xml version='1.0' encoding='UTF-8'?>语句之前不能有任何字符，包括空格和回车符。

（4）编写数据表 tb_message 所对应的持久化类 TbMessage。该类符合 JavaBean 规范，封装了对象的属性信息，并提供与其对应的 setXXX()和 getXXX()方法。关键代码如下：

```
public class TbMessage {
    private Integer id;                                        //ID 号
    private String writer;                                     //留言人
    private String content;                                    //留言内容
    private Timestamp sendTime;                                //留言时间
   public Integer getId() {
       return this.id;
   }
   public void setId(Integer id) {
       this.id = id;
   }
   //此处省略了其他属性对应的 setXXX()和 getXXX()方法
}
```

（5）创建持久化类 TbMessage 所对应的映射文件 TbMessage.hbm.xml，负责建立持久化类中的属性与数据表的字段之间的映射关系。需要注意的是，该文件应该与持久化类 TbMessage 在同一目录下。关键代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.wgh.model.TbMessage" table="tb_message" catalog="db_database16">
    <id name="id" column="id" type="int">
      <generator class="increment"/>   //设置 ID 字段自动增值
    </id>
        <property name="writer" type="java.lang.String">
            <column name="writer" length="45" not-null="true">
                <comment>留言人</comment>
            </column>
        </property>
        <property name="content" type="java.lang.String">
            <column name="content" length="200" not-null="true">
                <comment>留言内容</comment>
            </column>
        </property>
```

```
        <property name="sendTime" type="java.sql.Timestamp">
            <column name="sendTime" length="19" not-null="false">
                <comment>留言时间</comment>
            </column>
        </property>
    </class>
</hibernate-mapping>
```

说明： 在上面的代码中，<class>元素的 name 属性用于指定对应的持久化类；table 属性用于指定对应的数据表；catalog 属性用于指定对应数据表所在的数据库。

（6）编写 HibernateUtil 类，用于构建 SessionFactory，并通过 HibernateAPI 操作数据库。在该类中首先创建 SessionFactory，然后编写开启 Session 的方法，再编写调用 Session 的 save()方法保存留言信息的方法，最后编写关闭 Session 的方法。关键代码如下：

```
public class HibernateUtil {
    static SessionFactory sessionFactory;
    private Session session=null;
    private Transaction tx =null;
    //初始化 Hibernate，创建 SessionFactory 实例，只在该类被加载到内存时执行一次
    static{
        try{
            Configuration config = new Configuration().configure();
             sessionFactory = config.buildSessionFactory();            //构建 SessionFactory
        } catch (Exception e) {
             System.out.println("static 块中："+e.getMessage());
        }
    }
    //开启 Session
    public void openSession() {
        session = sessionFactory.openSession();
        tx = session.beginTransaction();                               //开启事务
    }
    //保存留言信息
    public String saveMessage(TbMessage message){
      try{
      openSession();                                                   //开启 Session
      session.save(message);                                           //调用 save()方法将留言信息保存到数据库
      tx.commit();                                                     //提交事务
      closeSession();                                                  //关闭 Session
      return "留言信息保存成功！";
      }catch(Exception e){
            e.printStackTrace();

            return "保存留言信息失败！";
      }
    }
    //关闭 Session
    public void closeSession() {
        session.close();
    }
}
```

（7）创建一个名为 MessageServlet 的类，用于完成业务逻辑处理。在该类中通过 doPost()方法接收留言信息，然后封装成 TbMessage 对象，并调用 HibernateUtil 类的 saveMessage()方法将留言信息保存到数据库中。关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)    throws ServletException, IOException {
      response.setCharacterEncoding("GBK");
      request.setCharacterEncoding("GBK");
      String writer=request.getParameter("writer");                    //获取留言人
      String content=request.getParameter("content");                  //获取留言内容
      HibernateUtil hibernateUtil=new HibernateUtil();
      TbMessage message = new TbMessage();                             //实例化对象
      message.setWriter(writer);                                       //设置留言人属性的值
      message.setContent(content);                                     //设置留言内容属性的值
      String returnStr=hibernateUtil.saveMessage(message);             //保存留言信息
```

```
        //弹出提示信息并重定向页面
        PrintWriter out = response.getWriter();
        out.print("<script>alert('"+returnStr+"');window.location.href='forward.jsp';</script>");
        out.flush();
        out.close();
}
```

（8）编写填写留言的信息页面 index.jsp，在该页面中添加用于收集留言的表单及表单元素。关键代码如下：

```
<form name="form1" method="post" action="MessageServlet" onSubmit="return check();">
  留言人：  <input name="writer" type="text" id="writer"> *
  留言内容：<textarea name="content" cols="70" rows="9" class="wenbenkuang" id="content"></textarea>   *
  <input name="Submit" type="submit" class="btn_bg" value="提交">
  <input name="Submit2" type="reset" class="btn_bg" value="重置">
</form>
```

秘笈心法

心法领悟 420：让数据库自动插入当前系统时间。

在实现添加留言信息时，最好让数据库自动插入发表留言的时间，这样能准确一些。在 MySQL 数据库中，可以通过将字段类型设置为 TIMESTAMP，将默认值设置为 CURRENT_TIMESTAMP 来实现。这时，再应用 Hibernate 向数据库中插入数据时，无须设置该字段的值。不过值得注意的是，在持久化类对应的映射文件中，需要将该字段的 not-null 属性设置为 false，而不能是 true。如果是 true，在添加数据时将产生以下异常信息。

```
org.hibernate.PropertyValueException:
 not-null property references a null or transient value: com.wgh.model.TbMessage.sendTime
```

实例 421　更新实体对象

光盘位置：光盘\MR\16\421

高级

实用指数：★★★★

实例说明

修改和更新数据在数据库操作中经常用到。在应用 JDBC 编程时，可以通过 SQL 语句来实现；但如果应用 Hibernate，就不需要通过 SQL 语句实现。Hibernate 提供了 update()方法，可以很方便地实现更新数据的功能。本实例将应用 update()方法对留言信息进行修改。运行本实例，将显示一个留言信息列表，如图 16.3 所示。单击要修改的留言信息后面的“修改”超链接，将进入如图 16.4 所示的修改留言信息页面。在该页面中，默认显示了要修改信息的原始值，修改信息后，单击“提交”按钮，将保存更新后的信息。

留言人	留言内容	修改
明日	明日科技编程词典	修改
wgh	测试111	修改
风铃	让风拨动我的心灵！	修改
无语	愿睿睿健康快乐！	修改
wgh	愿琦琦健康快乐！	修改
琦琦	琦琦	修改
无语	平安	修改
琦琦	开心	修改
wgh	test	修改

图 16.3　留言信息列表

留言人：明日 *

留言内容：明日科技编程词典 *

提　交　　重　置　　返　回

图 16.4　修改留言信息页面

本实例主要应用 update()方法实现更新数据的功能。update()方法的语法如下：

```
session.update(Object)
```

- Session：指的是 Session 实例，可以通过以下代码创建。

```
Configuration config = new Configuration().configure();
SessionFactory sessionFactory = config.buildSessionFactory();
Session session = sessionFactory.openSession();
```

- Object：用于指定持久化类的对象，即指定将哪个持久化对象保存到数据库中。

说明：在 Hibernate 中，更新数据的方法与保存数据的方法存在很大的差别。由于 Hibernate 缓存的存在及 Hibernate 的内部机制，更新数据首先要加载数据，然后再调用 update()方法对加载的数据进行更新，否则可能产生异常或数据不同步的情况。

（1）在 MyEclipse 中创建项目并导入 Hibernate 包。

（2）创建 Hibernate 配置文件 hibernate.cfg.xml。由于本实例与实例 420 使用的是同一个数据库及数据表，所以该实例的 hibernate.cfg.xml 与实例 420 相同，这里不再赘述。

（3）编写数据表 tb_message 所对应的持久化类 TbMessage 及对应的映射文件 TbMessage.hbm.xml。

（4）编写 HibernateUtil 类，用于构建 SessionFactory，并通过 HibernateAPI 操作数据库。在该类中首先创建 SessionFactory，然后编写开启 Session 的方法，再编写获取留言信息列表的方法 listMessage()、获取指定留言信息的方法 getMessage()和修改留言信息的方法 updateMessage()，最后编写关闭 Session 的方法。关键代码如下：

```
//获取留言信息列表
 public List<TbMessage> listMessage(){
      openSession();                                                    //开启 Session
       String hql="FROM TbMessage m ORDER BY m.sendTime DESC";          //降序查询全部留言信息
          List<TbMessage> list=null;
          try{
               Query query=session.createQuery(hql);
               list=(List<TbMessage>)query.list();
          }catch(Exception e){
               System.out.println("查询时的错误信息："+e.getMessage());
          }finally{
               session.close();
          }
          return list;
   }
   //获取指定留言信息
   public TbMessage getMessage(int id){
      openSession();                                                    //开启 Session
      TbMessage tbMessage=(TbMessage)session.get(TbMessage.class, id);  //通过 get()方法查询指定 ID 的留言信息
      session.close();                                                  //关闭 Session
      return tbMessage;
   }

   //修改留言信息
   public String updateMessage(TbMessage message){
      try{
            openSession();                                              //开启 Session
            //在应用 update()方法时，应该先调用 get()方法加载数据，然后再调用 update()方法更新数据
            TbMessage m=(TbMessage)session.get(TbMessage.class,message.getId());
            m.setWriter(message.getWriter());
            m.setContent(message.getContent());
            session.update(m);                                          //应用 update()方法修改留言信息到数据库
            tx.commit();                                                //提交事务
            closeSession();                                             //关闭 Session
            return "留言信息修改成功！";
      }catch(Exception e){
            e.printStackTrace();
            tx.rollback();                                              //事务回滚
            return "修改留言信息失败！";
      }
   }
```

说明：在 updateMessage()方法中，需要先调用 Session 的 get()方法获取要修改的留言信息，再调用 update()方法更新数据；否则，在修改留言信息后，发表留言的时间也将被修改。

（5）创建一个名为 MessageServlet 的 Servlet 类，在该类中根据传递的 action 参数的值执行不同的方法，从

而完成业务逻辑处理（在该类中，将调用步骤（4）中创建的 listMessage()、getMessage()和 updateMessage()方法实现显示留言列表和修改留言信息等功能）。

（6）编写显示留言列表的页面 listMessage.jsp 和显示要修改的留言信息的页面 showMessage.jsp。

秘笈心法

心法领悟 421：Hibernate 的 DTD 文件。

Hibernate 的 DTD 是非常复杂的，通常在编译器或者 IDE 中使用 DTD 来自动完成那些用来映射的 XML 元素和属性。可以通过在文本编译器里打开 DTD（这是最简单的方式）来概览所有的元素和 attribute，并查看其默认值以及注释。DTD 文件已包含在 Hibernate3.jar 中，同时在 Hibernate 发布包的 src 目录下也可找到。

实例 422 删除数据

光盘位置：光盘\MR\16\422

高级

实用指数：★★★★

实例说明

在 Hibernate 框架中，对数据的删除操作与对数据的修改比较类似，都需要先将数据加载，然后再对其进行相应的操作。对于批量删除数据，可以采用 Hibernate 提供的 HQL 查询语言或 Session 接口的 delete()方法。本实例中，将使用 Session 接口的 delete()方法对数据进行批量删除操作。运行本实例，选中左侧的复选框，单击“删除”按钮，即可对选择的学生信息进行删除，如图 16.5 所示。



图 16.5　批量删除学生信息

关键技术

本实例首先对持久化对象的主键 ID 进行收集，然后确定持久化对象并对其进行批量删除操作。这一操作过程主要涉及 Session 接口的 load()方法及 delete()方法，下面分别进行介绍。

（1）加载数据

对于还没有删除的对象，它处于 detached 状态，并没有纳入 Session 的管理之中，在删除操作前需要对数据进行加载。本实例中采用了 load()方法加载数据，该方法将返回持久化对象。语法如下：

```
public Object load(Class theClazz, Serializable id) throws HibernateException
```

参数说明

❶ theClazz：持久化类的.class 对象。

❷ id：持久化类中的标识对象。

注意： load()方法与 get()方法类似，其区别在于 load()方法采用了延迟加载，调用此方法返回的是代理对象，只有真正用到对象时，Hibernate 才会发出 SQL 语句去加载对象；而 get()方法调用将返回对象的实例。

（2）删除数据

Session 接口的 delete()方法用于删除对象。语法如下：

```
public void delete(Object object) throws HibernateException
```

参数说明

object：持久化类对象。

（1）导入 Hibernate 包及 Hibernate 的配置文件 hibernate.cfg.xml，在该配置文件中添加映射文件。

（2）创建持久化类 Student 及其相对应的映射文件 Student.hbm.xml。Student 类的关键代码如下：

```
public class Student {
	private Integer id;				//ID 编号
	private String stuName;			//姓名
	private Integer age;				//年龄
	private boolean sex;				//性别
	private String stuClass;			//班级
	private String description;		//描述信息
	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	//省略 get()和 set()方法
}
```

映射文件 Student.hbm.xml 的主键生成策略采用 native，其关键代码如下：

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
	"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
	"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
	<class name="com.lyq.vo.Student" table="tb_student">
		<id name="id">
			<generator class="native"/>
		</id>
		<property name="stuName" not-null="true" length="50"/>
		<property name="age"/>
		<property name="sex" type="boolean">
			<column name="sex" sql-type="int"/>
		</property>
		<property name="stuClass"/>
		<property name="description"/>
	</class>
</hibernate-mapping>
```

Student 类中的 sex 属性为布尔数据类型，在映射文件中通过 sql-type 属性将其转换为 int 型。

（3）创建名为 HibernateUtil 的类，用于操作 SessionFactory 及 Session 对象。

（4）创建名为 StuDao 的类，它是程序中的核心类，用于封装对数据库的操作。在此类中，编写 deleteStudent()方法用于批量添加数据。其关键代码如下：

```
public void deleteStudent(String[] ids){
	Session session = null;
	try {
		session = HibernateUtils.getSession();				//获取 Session
		session.beginTransaction();						//开启事务
		for (String s : ids) {							//通过循环获取主键 id
			Integer id = Integer.valueOf(s);				//转换为 Integer 型
			//通过 load()方法加载数据
			Student stu = (Student)session.load(Student.class, id);
			session.delete(stu);						//删除数据
		}
		session.getTransaction().commit();				//提交事务
	} catch (Exception e) {
		e.printStackTrace();							//打印异常信息
		session.getTransaction().rollback();			//回滚事务
	}finally{
		HibernateUtils.closeSession(session);			//关闭 Session
	}
}
```

（5）创建名为 StuServlet 的类（它是一个 Servlet），在此类中使用 doPost()方法对业务逻辑进行处理。首先

定义 String 类型的 command 参数，通过传入值进行业务逻辑的判断，并调用与其对应的方法对学生信息进行查询和删除操作。其关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    String command = request.getParameter("command");
    StuDao dao = new StuDao();
    if ("find".equals(command)) {                    //查询所有学生信息
        //查询所有学生信息
        List<Student> list = dao.findAllStudent();
        request.setAttribute("list", list);
        request.getRequestDispatcher("stu_list.jsp").forward(request,response);
    } else if ("delete".equals(command)) {           //批量删除学生信息
        //获取学生 id
        String[] ids = request.getParameterValues("id");
        if(ids != null && ids.length > 0){
            dao.deleteStudent(ids);                  //批量删除学生信息
        }
        request.getRequestDispatcher("StuServlet?command=find").forward(request,response);
    }
}
```

（6）创建页面 stu_list.jsp，用于显示学生信息。

秘笈心法

心法领悟 422：使用方法访问的必要性。

在设计 Hibernate 持久化类时，通常每个属性都包含有 getXXX()与 setXXX()方法，在此使用了标准 JavaBean 的命名约定，这是一种推荐的设计，但并不是必需的，Hibernate 也可以直接访问属性。使用 setXXX()与 getXXX()方法的优点是提供了重构时的健壮性。

实例 423	批量添加数据 光盘位置：光盘\MR\16\423	中级 实用指数：★★★★☆

实例说明

批量添加数据在项目中经常会用到，它可以一次性将数据写入数据库中，避免了每次添加数据都要对数据库进行频繁的开启和关闭等操作，从而节省了资源，大幅提高了数据库的效率。使用 Hibernate 可以轻松、快捷地批量添加数据，如本实例中通过 Hibernate 将图书信息批量添加到数据库中。运行本实例，在如图 16.6 所示的页面中填写图书信息，单击“添加”按钮，图书信息将以列表的方式显现在页面下方；添加完毕后，单击下方的“保存到数据库”按钮，图书信息将批量保存到数据库中。

图 16.6　批量添加图书信息

关键技术

本实例中，首先将批量添加的数据保存到 List 集合中，在添加数据之前开启 Session，然后通过 Session 接口的 save()方法依次将对象持久化到数据库中，最后关闭 Session 对象，从而做到省时省力。

设计过程

（1）导入 Hibernate 包。

（2）配置 Hibernate 配置文件 hibernate.cfg.xml，在该配置文件中添加映射文件。关键代码如下：

```
<session-factory>
	//Hibernate 方言
	<property name="dialect">org.hibernate.dialect.MySQLDialect</property>
	//数据库连接
	<property name="connection.url">jdbc:mysql://localhost:3306/db_database18</property>
	//数据库连接用户名
	<property name="connection.username">root</property>
	//数据库连接密码
	<property name="connection.password">111</property>
	//驱动
	<property name="connection.driver_class">com.mysql.jdbc.Driver</property>
	//显示 SQL 语句
	<property name="show_sql">true</property>
	//自动建表
	<property name="hibernate.hbm2ddl.auto">update</property>
	//映射文件
	<mapping resource="com/lyq/vo/Book.hbm.xml"/>
</session-factory>
```

（3）创建持久化类 Book 及其相对应的映射文件 Book.hbm.xml。关键代码如下：

```
public class Book {
	private Integer id;				//ID 编号
	private String bookName;		//图书名称
	private Double price;			//图书价格
	private Integer bookCount;		//图书数量
	private String category;		//图书类别
	private String description;		//图书描述信息
	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	//省略 get()和 set()方法
}
```

映射文件 Book.hbm.xml 的主键生成策略为 native 自动生成，其关键代码如下：

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
	"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
	"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
	<class name="com.lyq.vo.Book" table="tb_book">
		<id name="id">
			<generator class="native"/>
		</id>
		<property name="bookName" not-null="true" length="50"/>
		<property name="price" not-null="true"/>
		<property name="bookCount"/>
		<property name="category"/>
		<property name="description"/>
	</class>
</hibernate-mapping>
```

映射文件中的<property>元素除程序中所见到的属性外，还有其他很多属性，可根据实际情况进行设置。例如，经常用到的还有 column 属性、type 属性等，column 属性为表中字段名，type 属性为表中字段的类型。不设置这两个属性，Hibernate 将默认为 name 属性的值及持久化类中相对应的类型。由于本实例中没有特殊字段和类型，所以程序中采取默认方式。

（4）创建名为 HibernateUtil 的类，用于操作 SessionFactory 及 Session 对象。

（5）创建名为 BookDao 的类，它是程序中的核心类，用于封装对数据库的操作。此类中的 saveAllBooks()方法用于批量添加数据，其关键代码如下：

```
public void saveAllBooks(List<Book> books){
	Session session = null;
	if(books != null && books.size() > 0){
```

```
        try {
            session = HibernateUtils.getSession();                 //获取 Session
            session.beginTransaction();                            //开启事务
            Book book = null;                                      //创建 Book 对象
            for (int i = 0; i < books.size(); i++) {
                book = (Book)books.get(i);                         //获取 Book
                session.save(book);                                //保存对象
            }
            session.getTransaction().commit();                     //提交事务
        } catch (Exception e) {
            e.printStackTrace();                                   //打印错误信息
            session.getTransaction().rollback();                   //出错将回滚事务
        }finally{
            HibernateUtils.closeSession(session);                  //关闭 Session
        }
    }
}
```

（6）创建名为 BookServlet 的类，它是一个 Servlet，用于对业务逻辑进行处理。在此类中定义 String 类型的 command 参数，当传入参数为 add 时，将单条数据添加到 List 集合中，然后放入 Session 会话中；当传入参数为 save 时，将调用 BookDao 类的 saveAllBooks()方法批量保存数据。其关键代码如下：

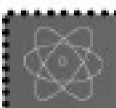
```
String command = request.getParameter("command");
//获取 Session
HttpSession session = request.getSession();
//从 Session 中获取已保存的图书
List<Book> list = (List)session.getAttribute("books");
if("add".equals(command)){                                  //向 Session 中添加图书
    //收集图书信息
    String bookName = request.getParameter("bookName");
    String price = request.getParameter("price");
    String bookCount = request.getParameter("bookCount");
    String category = request.getParameter("category");
    String desc = request.getParameter("description");
    Book book = new Book();                                 //创建 Book 对象
    book.setBookName(bookName);
    book.setBookCount(Integer.valueOf(bookCount));
    book.setPrice(Double.valueOf(price));
    book.setCategory(category);
    book.setDescription(desc);
    if(list == null){
        list = new ArrayList<Book>();
    }
    list.add(book);
    //将数据保存到 Session 中
    session.setAttribute("books", list);
    request.getRequestDispatcher("index.jsp").forward(request, response);
}else if("save".equals(command)){                           //向数据库中批量保存图书
    String info = "没有图书要保存！";                        //结果信息
    if(list != null && list.size() > 0){
        BookDao dao = new BookDao();                        //创建 BookDao 对象
        dao.saveAllBooks(list);                             //批量保存数据
        session.removeAttribute("books");                   //将数据从 Session 中移除
        info = "所有图书保存成功！";
    }
    request.setAttribute("info", info);
    request.getRequestDispatcher("result.jsp").forward(request, response);
}
```

（7）创建页面 index.jsp（即程序的首页），用于放置添加的图书表单信息。

秘笈心法

心法领悟 423：在创建 JavaBean 持久化类时注意 SQL 保留字的使用。

因为持久化类的属性要与数据库中的字段一一对应，因此 JavaBean 属性不但要满足不是 Java 中的保留字，还要满足不是数据库中的保留字，这样才能保存程序的正确执行，避免不必要的麻烦。



实例 424

采用一对一关联添加数据

光盘位置：光盘\MR\16\424

高级

实用指数：★★★★

实例说明



图 16.7　录入用户档案

在开发软件的过程中，经常会遇到一张数据表中的记录与另一张数据表中的记录一一对应的情况，即一对一的关系。Hibernate 提供了处理这种关系的映射方法，分为一对一主键关联映射和一对一外键关联映射，本实例中采用的是一对一外键关联映射（分别将个人基本信息与证件信息存放在两张表中）。运行本实例，在如图 16.7 所示页面中输入相关信息，然后单击“添加”按钮，档案就建立成功了。

关键技术

（1）在持久化类 User 中加入 IdCard 属性，在其映射文件 User.hbm.xml 中，通过<many-to-one>标签进行关系的映射。关键代码如下：

```
<many-to-one name="idCard" column="idCard" unique="true" not-null="true" cascade="all"/>
```

注意：<many-to-one> 标签主要用于多对一关联映射和一对一关联映射，本实例中通过使用 unique="true" 属性限制多重性，从而构成一对一的关系。

（2）在持久化类 IdCard 中加入 User 属性，在 IdCard.hbm.xml 映射文件中，通过<one-to-one>标签进行关系映射，并使用 property-ref 属性配置所映射的外键。关键代码如下：

```
<one-to-one name="user" property-ref="idCard"/>
```

（1）导入 Hibernate 包并配置 Hibernate 配置文件 hibernate.cfg.xml。

（2）创建持久化类 User 及 IdCard。因为它们的关系是双向关联的，各自持有对方的引用，所以要在 User 类中加入 IdCard 属性。关键代码如下：

```
private IdCard idCard;                                  //证件
public IdCard getIdCard() {
        return idCard;
}
public void setIdCard(IdCard idCard) {
        this.idCard = idCard;
}
```

在 IdCard 类中加入 User 属性，构造证件和用户的一对一双向关联关系。关键代码如下：

```
private User user;                                      //用户
public User getUser() {
        return user;
}
public void setUser(User user) {
        this.user = user;
}
```

（3）编写持久化类所对应的映射文件，其中用户的持久化类为 User.hbm.xml，通过<many-to-one>标签映射一对一关联关系。关键代码如下：

```
<hibernate-mapping package="com.lyq.vo">
        <class name="User" table="tb_user_025">
                <id name="id">
                        <generator class="native"/>
                </id>
```

```
            <property name="username" not-null="true" length="100"/>
            <property name="age" />
            <property name="sex" type="boolean">
                <column name="sex" sql-type="int"/>
            </property>
            <property name="nativePlace"/>
            <property name="address"/>
            <property name="description" type="text"/>
            <many-to-one name="idCard" column="idCard" unique="true" not-null="true" cascade="all"/>
        </class>
</hibernate-mapping>
```

证件的映射文件为 IdCard.hbm.xml，此文件中通过<one-to-one>标签映射一对一关联关系。关键代码如下：

```
<hibernate-mapping package="com.lyq.vo">
        <class name="IdCard" table="tb_idCard_025">
            <id name="id">
                <generator class="native"/>
            </id>
            <property name="type" not-null="true"/>
            <property name="num"/>
            <one-to-one name="user" property-ref="idCard"/>
        </class>
</hibernate-mapping>
```

（4）创建 UserDao 类，用于封装对数据库的操作。此类中提供了 saveUser()方法用于保存数据。由于配置了 User 对象与 IdCard 对象的级联关系，所以当保存 User 对象时，也将同时保存 IdCard 对象。关键代码如下：

```
public void saveUser(User user){
        try {
            session = HibernateUtils.getSession();          //获取 Session
            tx = session.beginTransaction();                //开启事务
            session.save(user);                             //保存 User 到数据库
            tx.commit();                                    //提交事务
        } catch (Exception e) {
            e.printStackTrace();                            //打印异常信息
            tx.rollback();                                  //回滚事务
        }finally{
            HibernateUtils.closeSession(session);           //关闭 Session
        }
}
```

（5）创建 UserServlet 类，它是一个 Servlet，用于处理 JSP 页面提交的请求。当录入用户信息时，将 IdCard 对象保存到 User 中。关键代码如下：

```
UserDao dao = new UserDao();                                //创建 UserDao
IdCard card = new IdCard();                                 //创建 IdCard
User user = new User();                                     //创建 User
//对 card 赋值
card.setType(type);
card.setNum(num);
//对 User 赋值
user.setUsername(username);
if(sex != null){
        user.setSex(sex.equals("1") ? true : false);
}
user.setAge(Integer.valueOf(age));
user.setNativePlace(nativePlace);
user.setAddress(address);
user.setDescription(desc);
user.setIdCard(card);                                       //设置用户所对应的 IdCard
dao.saveUser(user);                                         //级联保存 User
```

（6）创建程序的首页 index.jsp，放置录入的用户表单信息。

秘笈心法

心法领悟 424：Hibernate 线程绑定模式。

Session 在第一次被使用时，即第一次调用 getCurrentSession()方法时，其生命周期就开始了。然后被 Hibernate

绑定到当前线程。当事务结束时，不管是提交还是回滚，Hibernate 都会自动把 Session 从当前线程剥离，并将其关闭。假如程序再次调用 getCurrentSession()，将会得到一个新的 Session，并且开始一个新的工作单元。这种线程绑定的编程模式是 Hibernate 应用最广泛的方式之一。

实例 425　采用一对多关联添加数据

光盘位置：光盘\MR\16\425

中级　实用指数：★★★★

实例说明

在软件开发的过程中，经常会遇到一张数据表中的每条记录均与另一张数据表中的多条记录相对应的情况，即一对多的关系。Hibernate 提供了解决这种关系的简单方法，即建立一对多的关联关系。本例将以食品分类为例，分别将商品类别信息和商品信息存放在两张表中，演示一对多关联的使用方法。运行本实例，在如图 16.8 所示页面中输入商品类别名称及此类别中的多个商品，单击“提交”按钮，将对商品进行批量添加，并通过如图 16.9 所示页面显示所有商品信息。



图 16.8　添加商品信息页面

已添加商品

商品ID	商品类别	商品名称	单价	数量	产地
2	肉类	牛肉	18.00	20	长春
1	肉类	鸡肉	8.00	10	吉林
4	蔬菜	土豆	.80	100	农安
5	蔬菜	豆角	1.20	80	长春
3	蔬菜	白菜	1.00	100	长春

返回

图 16.9　已添加商品信息页面

关键技术

本实例通过建立商品类别与商品信息之间的一对多关联关系，使用 cascade 属性实现级联添加数据操作，其使用方法如表 16.1 所示。

表 16.1　cascade 属性的可选值及功能

控 件 类 型	控 件 命 名	控 件 用 途
JComboBox	priceComboBox	显示价格的排序方式
	stockComboBox	显示库存的排序方式
JButton	queryButton	显示“查询”按钮控件
JTable	table	显示查询结果的表格控件

（1）在 Hibernate 的一对多关联关系中，在“一”的一端通过 Set 集合装载对方的引用。如在本实例中持久化类商品类别中加入 merchs 属性，其数据类型为 Set 类型。关键代码如下：

```
private Set<Merch> merchs;                    //商品集合
public Set<Merch> getMerchs() {
        return merchs;
}
public void setMerchs(Set<Merch> merchs) {
        this.merchs = merchs;
}
```

（2）在映射文件中通过<set>标签配置一对多关系映射，并配置 cascade 属性设置级联操作类型。本实例中将 cascade 设置为 all（由于只是对其进行保存操作，也可设置为 save-update）。关键代码如下：

```
<set name="merchs" cascade="all" lazy="false">
        <key column="sortId"/>
        <one-to-many class="com.lyq.vo.Merch"/>
</set>
```

<set>元素用于一对多关联映射，其 name 属性要与持久化类对应的属性名称相一致，cascade 属性用于设置

级联操作类型；<key>元素的 column 属性值为与它关联的表的外键；<one-to-many>元素用于指定目标对象，它通过 class 属性设置一对多关联的实体对象。

设计过程

（1）导入 Hibernate 包，配置 Hibernate 文件 hibernate.cfg.xml，并在该配置文件中添加映射文件。

（2）创建商品信息持久化类 Merch 类、商品类别持久化类 MerchSort 类。由于它们存在一对多的关联关系，所以在 MerchSort 类中加入 Set 类型的属性 merchs。关键代码如下：

```
private Integer id;                //id 编号
private String name;               //类别名称
private Set<Merch> merchs;         //商品集合
public Set<Merch> getMerchs() {
        return merchs;
}
public void setMerchs(Set<Merch> merchs) {
        this.merchs = merchs;
}
```

（3）在持久化类映射文件中创建一对多关联映射关系，其中商品持久化类 Merch 的映射文件为 Merch.hbm.xml，在此映射文件中配置多对一关联映射关系。关键代码如下：

```
<many-to-one name="merchSort" column="sortId" lazy="false"/>
```

商品类别持久化类 MerchSort 的映射文件为 MerchSort.hbm.xml，它为“一”的一端，使用<set>标签配置一对多关系。关键代码如下：

```
<set name="merchs" cascade="all" lazy="false">
        <key column="sortId"/>
        <one-to-many class="com.lyq.vo.Merch"/>
</set>
```

（4）创建名为 MerchDao 的类，用于封装对数据库的操作。在此类中，使用 Session 的 save()方法保存商品类别数据，因为设置了级联操作，所以在保存商品类别时会同时对商品信息进行持久化。关键代码如下：

```
public void saveMerchSort(MerchSort ms){
        try {
                session = HibernateUtils.getSession();          //获取 Session
                tx = session.beginTransaction();                //开启事务
                session.save(ms);                               //保存商品类别
                tx.commit();                                    //提交事务
        } catch (Exception e) {
                e.printStackTrace();                            //打印异常信息
                tx.rollback();                                  //异常回滚事务
        }finally{
                HibernateUtils.closeSession(session);           //关闭 Session
        }
}
```

（5）创建 MerchServlet 类，用于对业务逻辑进行处理（此类是一个 Servlet，它通过表单提交的数据对 Merch 对象和 MerchSort 对象进行封装）；然后调用 MerchDao 类的 saveMerchSort()方法进行持久化数据。关键代码如下：

```
MerchSort ms = new MerchSort();                         //创建 MerchSort
ms.setName(sortName);                                   //为商品类别赋值
Set<Merch> set = new HashSet<Merch>();                  //创建 Set 对象
//通过循环批量保存商品及类别
for (int i = 0; i < name.length ; i ++) {
        Merch m = new Merch();                          //创建 Merch
        //为商品赋值
        m.setName(name[i]);
        m.setIntroduce(introduce[i]);
        m.setMerchCount(Integer.valueOf(merchCount[i]));
        m.setPrice(Double.valueOf(price[i]));
        set.add(m);                                     //将商品添加到 Set 集合中
}
ms.setMerchs(set);                                      //将所有商品保存到类别中
dao.saveMerchSort(ms);                                  //级联保存类别和商品
```

（6）创建 index.jsp 页面（即程序的首页），用于放置添加的商品信息表单。

秘笈心法

心法领悟 425：Hibernate 映射类型中的字符串类型。

在配置 Hibernate 映射文件时，需要注意的是，如果数据库的字段类型是 varchar 类型，对应的持久化类是 String 类型，在配置映射类型时要使用 string 类型，而不是 String。初学 Hibernate 的用户经常会搞混，一定要注意。

16.2　HQL 与 QBC 检索方式

实例 426　分组统计

光盘位置：光盘\MR\16\426

高级

实用指数：★★★☆

实例说明

分组统计是十分重要的查询语句，它根据某一列属性对记录进行分组，在对数据进行统计操作时经常会用到。HQL 查询语言同样支持数据的分组。本实例将通过 HQL 查询语言对商品销售数据按员工姓名进行分组统计，并查询员工销售总额。实例运行结果如图 16.10 所示。

个人销量统计

查看全部　　个人销量统计

员工	总量	总额
小井	95	85,600.00 元
小李	43	94,600.00 元
小刘	93	29,760.00 元
小王	18	18,000.00 元
小张	15	45,000.00 元

图 16.10　分组统计

本实例通过 HQL 查询语言对商品销售信息进行分组查询，商品销售表数据如图 16.11 所示。

id	name	operator	totalPrice	totalCount	month
1	XXX牌电脑	小李	44000	20	2009-1
2	XXX牌显示器	小王	18000	18	2009-1
3	XXX牌显示器	小井	15000	15	2009-2
4	XXX牌打印机	小张	21000	7	2009-2
5	XXX牌光驱	小刘	16000	50	2009-2
6	XXX牌光驱	小刘	13760	43	2009-1
7	XXX牌打印机	小张	24000	8	2009-1
8	XXX牌显示器	小井	20000	20	2009-1
9	XXX牌电脑	小李	50600	23	2009-2
10	XXX牌显示器	小井	50600	60	2009-2

图 16.11　商品销售表

与 SQL 相同，HQL 查询语言使用关键字 GROUP BY 进行分组（可以通过关键字 HAVING 对分组条件进行限制）。其使用要点如下：

（1）编写使用分组进行统计查询的 HQL 语句，关键代码如下：

```
select s.operator,sum(totalCount),sum(totalPrice) from SellMonth s group by s.operator
```

注意：HQL 查询语言是面向对象查询语言，支持多态查询，在查询对象中的属性时，使用“.”明确区分对象的属性，如 group by s.operator。

（2）使用 Session 接口的 createQuery()方法创建 Query 对象，并通过 Query 接口的 list()方法获取结果集。关键代码如下：

```
list = session.createQuery(hql)             //创建 Query 对象
.list();                                    //获取结果集
```

设计过程

（1）导入 Hibernate 包及配置 Hibernate 配置文件 hibernate.cfg.xml。

（2）创建商品销售信息持久化类 SellMonth 及相对应的映射文件 SellMonth.hbm.xml。

（3）创建名为 SellDao 的类，用于封装对数据库的操作。在此类中编写 findGroupByOperator()方法，对商品销售信息按员工进行分组统计。关键代码如下：

```
public List<SellMonth> findGroupByOperator(){
        List<SellMonth> list = null;
        try {
                session = HibernateUtils.getSession();                  //获取 Session
                tx = session.beginTransaction();                        //开启事务
                //HQL 分组查询语句
                String hql = "select s.operator,sum(totalCount),sum(totalPrice) from SellMonth s group by s.operator";
                list = session.createQuery(hql)                         //创建 Query 对象
                                .list();                                //获取结果集
                tx.commit();                                            //提交事务
        } catch (Exception e) {
                e.printStackTrace();                                    //打印异常信息
        }finally{
                HibernateUtils.closeSession(session);                   //关闭 Session
        }
        return list;
}
```

（4）创建名为 SellServlet 的类，它是一个 Servlet，用于对业务请求进行处理。当请求查询分组统计信息时，将调用 SellDao 类中的 findGroupByOperator()进行查询，并将结果集信息转发到 sell_group.jsp 页面予以显示。

（5）创建 sell_list.jsp 页面，用于显示所有商品信息。

（6）创建 sell_group.jsp 页面，用于显示分组统计后的结果信息。当使用 GROUP BY 进行分组后，Query 接口所返回的 List 集合中，装载的并不是已封装的持久化类对象，而是 Object 数组。实例中通过 JSTL 标签获取。关键代码如下：

```
<c:forEach items="${list}" var="p">
        <tr>
                <td>${p[0]}</td>
                <td>${p[1]}</td>
                <td>
                        <fmt:formatNumber value="${p[2]}" pattern="#,###.00"/>元
                </td>
        </tr>
</c:forEach>
```

秘笈心法

心法领悟 426：持久化实例的 3 种状态。

Hibernate 的持久化实例有 3 种状态：游离状态，即实例从未与任何持久化上下文关联过；持久化状态，这样的对象拥有持久化标识，并且可能在数据库中有一个对应的行；脱管状态，这样的持久化实例曾经与某个持久化上下文发生过关联，不过那个上下文被关闭了，或者这个实例是被序列化到另外的进程。

实例 427	利用统计函数 SUM 求销售总额 光盘位置：光盘\MR\16\427	高级 实用指数：★★★★

实例说明

HQL 查询语言不仅支持对数据的分组操作，还支持对聚合函数的使用。在 HQL 查询语言中，支持大部分 SQL 中使用的聚合函数。本实例将使用聚合函数 sum()查询商品销量总额，运行结果如图 16.12 所示。

当前位置：基础信息 > 图书管理 >>>　2009年03月30日 星期一 13:21:45

销售额统计

商品名称	数量	总额
Java 编程词典	5	1,500.00元
Java 范例宝典	5	495.00元
SQL Server 编程词典	7	1,820.00元
Visual Basic编程词典	9	2,070.00元
Visual C++编程词典	3	720.00元

总计：6,605.00元　　返回

CopyRight 2009 www.mingrisoft.com 吉林省明日科技有限公司

图 16.12　查看销售总额

关键技术

聚合函数 sum()在 HQL 中的使用方法与 SQL 类似。本实例将对商品销售信息进行查询，统计销售总额。可通过聚合函数 sum()来实现，其 HQL 查询代码如下：

```
select sum(price) from Produce
```

（1）导入 Hibernate 包及 Hibernate 的配置文件 hibernate.cfg.xml。

（2）创建商品销售信息的持久化类 Produce 及相对应的映射文件 Produce.hbm.xml。

（3）创建名为 ProduceDao 的类，用于封装对数据库的操作。在此类中，编写查询销售总额的方法 findSumPrice()。关键代码如下：

```
public double findSumPrice(){
        double sum = 0;                                          //销售总额
        try {
                List<Produce> list = null;                       //结果集
                session = HibernateUtils.getSession();           //获取 Session
                tx = session.beginTransaction();                 //开启事务
                String hql = "select sum(price) from Produce";
                list = session.createQuery(hql).list();
                if(list != null && list.size() > 0){
                Object o = list.get(0);                          //获取销售总额
                        sum = Double.valueOf(o.toString());      //将销售总额转换为 double 类型
                }
                tx.commit();                                     //提交事务
        } catch (Exception e) {
                e.printStackTrace();                             //打印异常信息
        }finally{
                HibernateUtils.closeSession(session);            //关闭 Session
        }
        return sum;
}
```

使用 sum()进行统计查询，所得到结果中的条目并不是持久化对象，而是 Object 类型，所以在实例中通过 Object 接收，然后转换成 double 类型。

（4）创建名为 ProduceServlet 的 Servlet 类，用于对业务请求进行控制。在此类中将通过 command 参数判断请求类型，当 command 值为 find 时，查询所有商品信息；当 command 值为 findSumPrice 时，查询商品的销售总额。关键代码如下：

```
String command = request.getParameter("command");
ProduceDao dao = new ProduceDao();
List<Produce> list = null;
if("find".equals(command)){                        //查询所有商品
        list = dao.findAllProduce();
        request.setAttribute("list", list);
```

```
        request.getRequestDispatcher("produce_list.jsp").forward(request, response);
}else if("findSumPrice".equals(command)){
        list = dao.findGroupByName();                    //分组统计
        double d = dao.findSumPrice();                   //查询总额
        if(d != 0){
                request.setAttribute("sumPrice", d);
        }
        request.setAttribute("list", list);
        request.getRequestDispatcher("produce_sum.jsp").forward(request, response);
}
```

（5）创建用于显示所有销售数据的页面 produce_list.jsp、查看销售总额的页面 produce_sum.jsp。

秘笈心法

心法领悟 427：HQL 查询语句。

查询语句对大小写并无要求，例如，sElECt 和 SelecT 及 select 都是一样的，但是 Java 类与属性的名称除外。

实例 428 利用统计函数 AVG 求某班学生的平均成绩

高级

光盘位置：光盘\MR\16\428

实用指数：★★★★

实例说明

在 HQL 中获取数据表中某一列的平均值，可使用聚合函数 avg()来实现。本实例将使用 HQL 查询语言，通过 avg()函数查询班级平均成绩。在如图 16.13 所示页面中选择指定班级后单击“查看”按钮，将显示班级平均成绩，如图 16.14 所示。

当前位置：基础信息 > 所有学生成绩 >>> 2009年03月29日 星期四 13:21:45

查看班级平均成绩：请选择班级 查看（下拉选项：请选择班级、一年一班、一年二班、所有班级）

姓名	年龄	性别	分数	班级	描述
小王	20	男	100		学习成绩优秀
小李	21	女	95	一年二班	学习成绩一般
小张	25	女	81.5	一年二班	爱好篮球
小明	22	男	90	一年一班	学习成绩一般
小红	20	女	92	一年二班	爱学习
小井	21	男	70	一年二班	不爱学习
小刘	20	女	99	一年一班	很聪明

图 16.13　所有学生成绩

当前位置：基础信息 > 所有学生成绩 >>> 2009年03月29日 星期四 13:21:45

班级平均分数

班级	总分数	人数	平均分
一年一班	289	3	96.33

返回

图 16.14　使用 Form.Element 对象返回特定表单域的值

关键技术

使用 HQL 查询某班学生的平均成绩，主要涉及了 GROUP BY 子句及聚合函数 avg()的使用。

（1）首先要对班级进行分组，然后通过 HAVING 关键字对班级条件进行设置。关键代码如下：

```
from Student s group by s.stuClass having s.stuClass = ?
```

此语句采用了 HQL 的动态参数赋值，“?”号代表参数，其使用方法与 JDBC 中 PreparedStatement 类的动态赋值方法相类似。

（2）使用 avg()函数获取平均值。本实例中获取结果集的关键代码如下：

```
hql = "select stuClass,sum(grade),count(*),avg(grade) from Student s group by s.stuClass having s.stuClass = ?";
list = session.createQuery(hql)             //创建 Query 对象
.setParameter(0, stuClass)                  //对参数赋值
.list();                                    //获取结果集
```

设计过程

（1）导入 Hibernate 包及 Hibernate 的配置文件 hibernate.cfg.xml。

（2）创建学生持久化类 Student 及相对应的映射文件 Student.hbm.xml。

（3）创建 StudentDao 类，用于封装对数据库的操作。在此类中编写查询班级平均成绩的方法 findAvgGradeByClass ()，其关键代码如下：

```
public List<Object> findAvgGradeByClass(String stuClass){
	List<Object> list = null;
	try {
		session = HibernateUtils.getSession();			//获取 Session
		tx = session.beginTransaction();				//开启事务
		String hql = "";							//HQL 语句
		if("all".equals(stuClass)){
			//查询所有班级学生平均成绩
			hql = "select stuClass,sum(grade),count(*),avg(grade) from Student s group by s.stuClass";
			list = session.createQuery(hql).list();
		}else{
			//查询指定班级学生平均成绩
			hql = "select stuClass,sum(grade),count(*),avg(grade) from Student s group by s.stuClass having s.stuClass = ?";
			list = session.createQuery(hql)				//创建 Query 对象
					.setParameter(0, stuClass)			//对参数赋值
			.list();								//获取结果集
		}
		tx.commit();								//提交事务
	} catch (Exception e) {
		e.printStackTrace();						//打印异常信息
	}finally{
		HibernateUtils.closeSession(session);			//关闭 Session
	}
	return list;
}
```

此方法的入口参数是 String 类型的 stuClass，代表查询的范围。当 stuClass 的值为 all 时，将查询所有班级学生平均成绩；否则，将只查询某一班级的平均成绩。

（4）创建名为 StudentServlet 的 Servlet 类，用于对业务请求进行控制。它通过调用 StudentDao 类中的相应方法返回结果信息，并通过 JSP 页面予以显示。

（5）创建用于显示所有学生信息的页面 student_list.jsp、显示班级平均成绩的页面 student_avg.jsp。由于在查询班级平均成绩时，所返回的结果集中的条目并不是已创建的持久化类对象，而是 Object[]型，所以输出班级平均成绩时，要以数组的方式输出。关键代码如下：

```
<tr>
	<td><b>班级</b></td>
	<td><b>总分数</b></td>
	<td><b>人数</b></td>
	<td><b>平均分</b></td>
</tr>
<c:forEach items="${list}" var="s">
	<tr>
		<td>${s[0]}</td>
		<td>
			<fmt:formatNumber pattern="##.##">${s[1]}</fmt:formatNumber>
		</td>
		<td>${s[2]}</td>
		<td>
			<fmt:formatNumber pattern="##.##">${s[3]}</fmt:formatNumber>
		</td>
	</tr>
</c:forEach>
```

秘笈心法

心法领悟 428：C3P0 连接池。

C3P0 是一个随 Hibernate 一同分发的开源的 JDBC 连接池，位于 lib 目录下。如果设置了 hibernate.c3p0.*相关的属性，Hibernate 将使用 C3P0ConnectionProvider 缓存 JDBC 连接。

实例 429 利用统计函数 COUNT 统计当前注册用户人数

光盘位置：光盘\MR\16\429

高级

实用指数：★★★

实例说明

统计函数 count()主要用于统计数据的记录数。无论是在分页技术中，还是普通查询中都经常被用到。在 HQL 查询语言中，同样支持此函数的使用，其使用方法与 SQL 相类似。本实例将使用 HQL 查询语言，通过函数 count()统计网站中已经注册的人数，运行结果如图 16.15 所示。

图 16.15　查询已注册人数

关键技术

本实例通过统计用户数据表中数据的记录数，确定网站中已经注册的人数。实现这一过程主要使用了聚合函数 count()及 Hibernate 的单值检索功能。

（1）聚合函数 count()

与 SQL 相同，在 HQL 中也可以使用 count()函数查询数据的记录数。本实例中采用 count(*)查询数据的记录数，其 HQL 语句如下：

```
select count(*) from User
```

（2）单值检索

Query 接口不仅提供了返回结果集的 list()方法，而且提供了返回单条数据的方法 uniqueResult()。此方法返回值为 Object 对象，它能保证返回的值为单个对象，当查询条件返回多条记录时，将抛出异常。

在本实例中，使用此方法获取注册人数。由于它返回 Object 类型，实例中将其转换为 Long 型。关键代码如下：

```
count = (Long)session.createQuery(hql).uniqueResult();
```

设计过程

（1）导入 Hibernate 包及 Hibernate 的配置文件 hibernate.cfg.xml。

（2）创建用户持久化类 User 及相对应的映射文件 User.hbm.xml。

（3）创建名为 UserDao 的类，用于封装对数据库的操作。在此类中编写 getCount()方法，用于查询已注册的人数，其返回值为 Long 型数值。关键代码如下：

```
public long getCount(){
        Long count = null;
        try {
                session = HibernateUtils.getSession();                  //获取 Session
                tx = session.beginTransaction();                        //开启事务
                String hql = "select count(*) from User";               //查询数量语句
                //单值检索（返回对象）
                count = (Long)session.createQuery(hql).uniqueResult();
                tx.commit();                                            //提交事务
        } catch (Exception e) {
                e.printStackTrace();                                    //打印异常信息
        }finally{
                HibernateUtils.closeSession(session);                   //关闭 Session
        }
        return count;
}
```

注意：采用聚合函数 count()查询数据记录数时，返回的对象为数值类型，在 Hibernate 3.2 中需要使用 Long 型数据接收，否则会抛出异常。

（4）创建名为 UserServlet 的 Servlet 类，此类中通过 doPost()方法对请求进行处理。它通过调用 UserDao 类的 findAllUsers()方法和 getCount()方法查询所有用户信息及注册人数，并将查询到的数据转发到 JSP 页面予以显示。关键代码如下：

```
UserDao dao = new UserDao();                          //创建 UserDao 对象
List<User> list = dao.findAllUsers();                 //查询所有注册用户
Long count = dao.getCount();                          //查询注册用户数量
request.setAttribute("list", list);                   //将用户集合放入 request 中
request.setAttribute("count", count);                 //将注册用户数量放入 request 中
//转发到 user_list.jsp 页面
request.getRequestDispatcher("user_list.jsp").forward(request, response);
```

（5）创建 JSP 页面 user_list.jsp，用于显示已注册用户及注册人数。

秘笈心法

心法领悟 429：Hibernate 的统计机制。

如果开启 hibernate.generate_statistics，那么当通过 SessionFactory.getStatistics()调整正在运行的系统时，Hibernate 将导出大量有用的数据。Hibernate 甚至能被配置成通过 JMX 导出这些统计信息。

实例 430　利用 HQL 查询图书表中的所有数据

光盘位置：光盘\MR\16\430

中级

实用指数：★★★★

实例说明

HQL 查询语言同样提供了排序功能，其使用方法与 SQL 类似。本实例将实现查询所有图书信息，并将查询结果进行排序，通过选择“排序字段”和“排序类型”下拉列表框中的值，单击“查询”按钮，即可得到排序后的结果集。运行结果如图 16.16 所示。

查看所有图书

At a sale

排序查询：排序字段　排序类型　查询

ID	图书名称	价格	数量	类别	描述信息
1	故事大全	10.00元	90	中小学图书	科幻故事。
2	作文1000篇	39.00元	100	中小学图书	中小学图书
3	散文集选	35.00元	60	文学图书	中学生散文集选
4	名师名讲100分	20.00元	40	中小学图书	小学三年级数学
5	JavaScript网页特效范例宝典	79.00元	80	计算机图书	适用于广大计算机爱好者和编程人员使用
6	Spring应用开发完全手册	59.00元	35	计算机图书	适合大中专院校师生学习参考

图 16.16　查看所有图书

关键技术

在 HQL 查询语言中，排序的方法与 SQL 相似，它也使用 order by 子句对查询结果集进行排序，并且可使用 asc 或 desc 关键字指定排序方式。在此要注意的是，HQL 是面向对象查询语言，要严格区分对象与属性的大小写。

例如，对图书信息按价格的升序排序，HQL 代码如下：

```
from Book b order by b.price asc
```

设计过程

（1）导入 Hibernate 包及 Hibernate 的配置文件 hibernate.cfg.xml。

（2）创建图书持久化类 Book 及相对应的映射文件 Book.hbm.xml。

（3）创建名为 BookDao 的类，用于封装对数据库的操作。在此类中分别编写 findAllOrderByPrice()方法、findAllOrderByCount()方法，用于对结果集进行不同的排序。其中，findAllOrderByPrice()方法通过价格对结果集进行排序，其关键代码如下：

```
//查询所有图书，按图书价格排序
public List<Book> findAllOrderByPrice(String type){
```

```
List<Book> list = null;
try {
        session = HibernateUtils.getSession();                    //获取 Session
        session.beginTransaction();                               //开启事务
        //HQL 语句
        String hql = "from Book b order by b.price asc";
        if("desc".equals(type)){
                hql = "from Book b order by b.price desc";
        }
        list = session.createQuery(hql).list();
        session.getTransaction().commit();                        //提交事务
} catch (Exception e) {
        e.printStackTrace();                                      //打印错误信息
}finally{
        HibernateUtils.closeSession(session);                     //关闭 Session
}
return list;
}
```

findAllOrderByCount()方法通过数量对结果集进行排序，其关键代码如下：

```
//查询所有图书，按图书数量排序
public List<Book> findAllOrderByCount(String type){
List<Book> list = null;
try {
        session = HibernateUtils.getSession();                    //获取 Session
        session.beginTransaction();                               //开启事务
        //HQL 语句
        String hql = "from Book b order by b.bookCount asc";
        if("desc".equals(type)){
                hql = "from Book b order by b.bookCount desc";
        }
        list = session.createQuery(hql).list();
        session.getTransaction().commit();                        //提交事务
} catch (Exception e) {
        e.printStackTrace();                                      //打印错误信息
}finally{
        HibernateUtils.closeSession(session);                     //关闭 Session
}
return list;
}
```

这两个方法的入口参数为 type，用于指定排序类型为升序还是降序。

（4）创建名为 BookServlet 的 Servlet 类，用于对业务逻辑进行控制。在此类中通过接收表单参数，调用 BookDao 类中的相应方法获取数据结果集，并转发到 book_list.jsp 页面予以显示。

（5）创建 index.jsp 页面，即程序的首页。

（6）创建 book_list.jsp 页面，用于显示查询的结果集。

秘笈心法

心法领悟 430：域和局部变量的区别。

除了域之外，还可以在方法和块中定义变量，这些变量称为局部变量。需要注意的是，对于局部变量，虚拟机并不会为其赋默认值。这意味着在使用这些变量前必须对其初始化。这是域和局部变量的重要区别。块是使用“{”和“}”包围的结构，可以用来实现初始化等功能。

实例 431 利用 HQL 查询满足指定条件的数据

中级

光盘位置：光盘\MR\16\431

实用指数：★★★★

实例说明

在 HQL 查询语言中，提供了多种按日期查询数据的方法。本实例将利用 between…and 子句实现按日期查

询销售明细。运行本实例，在如图 16.17 所示页面中正确输入起始日期和结束日期，单击“按日期查询”按钮，即可显示指定日期范围内的数据。

销售明细

起始日期：2009-01-21　结束日期：2009-02-21　按日期查询

注意：日期格式为：yyyy-mm-dd

ID	商品名称	总价	数量	销售日期	销售员
1	Visual Basic编程词典	690.00元	3	2009-02-06 01:33:04	小张
2	Java 编程词典	1,500.00元	5	2009-01-02 02:40:21	小王
3	SQL Server 编程词典	1,820.00元	7	2009-02-25 15:43:11	小李
4	Visual Basic编程词典	690.00元	3	2009-01-23 08:31:39	小张
5	Visual Basic编程词典	690.00元	3	2009-02-21 09:52:48	小李
6	Visual C++编程词典	720.00元	3	2009-02-22 09:18:21	小李
7	Java 范例宝典	495.00元	5	2009-01-02 04:08:21	小李

图 16.17　按日期查询销售明细

关键技术

HQL 查询语言为面向对象的查询语言。在书写 HQL 语句时，要严格区分大小写。HQL 与 SQL 语法相似，但其查询的目标为对象类型。在学习使用 HQL 检索方式检索数据之前，先来看一下 HQL 检索数据的步骤。

（1）编写 HQL 语句

HQL 语句与 SQL 非常相似，具有 SQL 语句中的常用关键字，如 select、from、where 等。在 HQL 中，如果查询对象的全部属性，可以省略 select 关键字。例如，本实例中查询 Produce 对象的全部属性。代码如下：

```
from Produce
```

使用此 HQL 语句，Hibernate 将发出类似于 select * from produce 的语句。

（2）创建 Query 对象

Query 对象用于执行 HQL 语句，通过 Session 接口的 createQuery()方法进行创建。语法如下：

```
public Query createQuery(String queryString) throws HibernateException
```

参数说明

queryString：HQL 查询语句。

（3）获取结果集

Query 接口的 list()方法用于获取查询的结果集，返回 List 集合。语法如下：

```
public List list() throws HibernateException
```

返回的 List 集合中装载的是所查询的持久化对象，对于这些对象，Hibernate 已经作出了赋值及封装，这是 HQL 查询方式的优点之一。

（1）导入 Hibernate 包及 Hibernate 的配置文件 hibernate.cfg.xml。

（2）创建商品持久化类 Produce 及相对应的映射文件 Produce.hbm.xml。

（3）创建 ProduceDao 类，用于封装对数据库的操作。在此类中编写两个方法，其中，findAllProduce()方法用于查询所有商品信息，其关键代码如下：

```
//查询所有数据
public List<Produce> findAllProduce(){
        List<Produce> list = null;
        try {
                session = HibernateUtils.getSession();                    //创建 Session 对象
                tx = session.beginTransaction();                          //开启事务
                list = session.createQuery("from Produce").list();        //查询 Produce 中所有数据
                tx.commit();                                              //提交事务
        } catch (Exception e) {
                e.printStackTrace();                                      //打印异常信息
        }finally{
                HibernateUtils.closeSession(session);                     //关闭 Session
        }
        return list;
}
```

findAllProduceByDate()方法用于查询指定日期范围内的商品信息。在此方法中，通过 between…and 子句指

定所查询的日期范围（与 SQL 类似）。关键代码如下：

```
//查询指定日期范围内的数据
public List<Produce> findAllProduceByDate(Date startTime, Date endTime){
	List<Produce> list = null;
	try {

		session = HibernateUtils.getSession();				//创建 Session 对象
		tx = session.beginTransaction();					//开启事务
		//查询指定日期范围内的数据
		list = session.createQuery("from Produce p where p.sellTime between ? and ?")
					.setParameter(0, startTime)
					.setParameter(1, endTime)
					.list();
		tx.commit();								//提交事务
	} catch (Exception e) {
		e.printStackTrace();							//打印异常信息
	}finally{
		HibernateUtils.closeSession(session);				//关闭 Session
	}
	return list;
}
```

在 HQL 语言中，可以用“?”代表参数，然后对其动态赋值。可以通过 Query 接口的 setParameter()方法实现，此种方法可避免 SQL 的注入式攻击。

注意： 由于 Query 接口的 setParameter()返回值仍然是 Query 对象，所以程序代码采取上面的方法进行书写，从而减少代码量。

（4）创建名为 ProduceServlet 的 Servlet 类，用于对业务逻辑进行处理。由于数据库中的销售日期为 dateTime 类型，而要查询数据的条件是根据日期查询，所以此类中通过 SimpleDateFormat 类构造了指定的日期范围。关键代码如下：

```
String startTime = request.getParameter("startTime");			//获取起始日期
String endTime = request.getParameter("endTime");			//获取结束日期
//创建 SimpleDateFormat 对象
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
startTime += " 00:00:00";								//精确起始日期
endTime += " 23:59:59";								//精确结束日期
try {
	//查询在起始日期和结束日期范围内的数据
	list = dao.findAllProduceByDate(sdf.parse(startTime), sdf.parse(endTime));
} catch (ParseException e) {
	e.printStackTrace();
}
```

秘笈心法

心法领悟 431：JNDI 绑定 SessionFactory。

JNDI 绑定的 Hibernate SessionFactory 可以简化工厂的查询，简化创建新的 Session。需要注意的是，这与 JNDI 绑定 Datasource 没有关系，它们只是恰巧用了相同的注册表。

实例 432　HQL 绑定参数查询　高级

光盘位置：光盘\MR\16\432　　实用指数：★★★★

HQL 查询语言的功能十分强大，提供了对数据模糊匹配的查询方法——模糊查询。在模糊查询中，其使用方法与 SQL 操作类似，通过关键字 like 和通配符进行模糊查询。

本实例将演示在商品销售表中通过商品名称的模糊匹配检索数据。在查询过程中查询条件是用户随意输入

的，如输入模糊匹配的商品名称，然后单击“模糊查询”按钮，即可进行模糊查询，如图 16.18 所示。

商 品 名 称：词典　模糊查询

销售明细

ID	商品名称	总价	数量	销售日期	销售员
1	Visual Basic编程词典	690.00元	3	2009-02-06 01:33:04	小张
2	Java 编程词典	1,500.00元	5	2009-01-02 02:40:21	小王
3	SQL Server 编程词典	1,820.00元	7	2009-02-25 15:43:11	小李
4	Visual Basic编程词典	690.00元	3	2009-01-23 08:31:39	小张
5	Visual Basic编程词典	690.00元	3	2009-02-21 09:52:48	小李
6	Visual C++编程词典	720.00元	3	2009-02-22 09:18:21	小李
7	Java 范例宝典	495.00元	5	2009-01-02 04:08:21	小李

图 16.18　模糊查询

关键技术

通配符“%”匹配的是 0 个或更多且任意长度的字符串，它在 HQL 查询语言中所代表的意义与 SQL 中相同。在本实例中，对商品名称的模糊查询将通过通配符“%”来实现。

（1）编写在商品信息表中对商品名称进行模糊查询的 HQL 语句。从安全方面着想，此语句采用了动态赋值方式。关键代码如下：

```
from Produce p where p.name like ?
```

（2）使用 Session 接口的 createQuery()方法创建 Query 对象，并对语句中的参数使用通配符组合进行动态赋值，从而获取结果集。关键代码如下：

```
list = session.createQuery("from Produce p where p.name like ?")          //创建 Query 对象
                .setParameter(0, "%" + s + "%")                           //使用通配符为参数赋值
                .list();                                                  //返回结果集
```

设计过程

（1）导入 Hibernate 包及 Hibernate 的配置文件 hibernate.cfg.xml。

（2）创建商品持久化类 Produce 及相对应的映射文件 Produce.hbm.xml。

（3）创建 ProduceDao 类，用于封装对数据库的操作。在此类中，编写通过商品名称进行模糊查询的方法 blurQuery()，其入口参数为 String 类型的模糊商品名称。关键代码如下：

```
public List<Produce> blurQuery(String s) {
        List<Produce> list = null;
        if (s != null && !s.isEmpty()) {
                try {
                        session = HibernateUtils.getSession();                   //获取 Session
                        tx = session.beginTransaction();                         //开启事务
                        //通过模糊匹配的商品名称进行模糊查询
                        list = session.createQuery("from Produce p where p.name like ?")
                                        .setParameter(0, "%" + s + "%")
                                .list();
                        tx.commit();                                             //提交事务
                } catch (Exception e) {
                        e.printStackTrace();                                     //打印异常信息
                }finally{
                        HibernateUtils.closeSession(session);                    //关闭 Session
                }
        }
        return list;
}
```

（4）创建名为 ProduceServlet 的 Servlet 类，它通过 doPost()方法对请求进行处理。在此方法中，将接收 JSP 页面传递的模糊商品名称，调用 ProduceDao 类中的 blurQuery()方法进行模糊查询，并将结果信息转发到 produce_list.jsp 页面予以显示。

秘笈心法

心法领悟 432：Query 和 Criteria 接口。

这两个接口都是查询接口，Query 封装了 HQL 查询语句，该语句是面向对象的，它引用类名以及类的属性名，而不是数据库中的表名和字段名；Criteria 接口封装了基于字符串形式的查询语句，比 Query 接口更面向对象，擅长执行动态查询。

实例 433　只返回一个检索对象

光盘位置：光盘\MR\16\433

高级

实用指数：★★★★

实例说明

强大的 HQL 查询语言几乎支持 SQL 的所有功能（除一些 SQL 特殊扩展外），如常用的聚合函数。本实例使用聚合函数 max()，实现查询月销售额完成最多的员工。选择要查询的月份，单击“查询”按钮，将显示所查询月份中销售额完成最多的员工，如图 16.19 所示。

ID	商品名称	销售总额	销售总量	员工	销售月份
1	XXX牌电脑	44,000.00元	20	小李	2009-1
2	XXX牌显示器	18,000.00元	18	小王	2009-1
3	XXX牌显示器	15,000.00元	15	小井	2009-2
4	XXX牌打印机	21,000.00元	7	小张	2009-2
5	XXX牌光驱	16,000.00元	50	小刘	2009-2

图 16.19　查询月销售额完成最多的员工

关键技术

本实例通过聚合函数 max()查询销售明细表中销售额完成最多的员工。其实现主要是确定 HQL 查询语句，可分为以下两步。

（1）查询出月份中最大的销售额，其 HQL 查询代码如下：

```
select max(totalPrice) from SellMonth s where s.month = ?
```

（2）当查询到月份中最大的销售额后，通过子查询查找月份中销售额等于这一数值的员工。其完整的 HQL 代码如下：

```
String hql = "from SellMonth s where s.month = ? and s.totalPrice = ";
hql += " (select max(totalPrice) from SellMonth s where s.month = ?)";
```

注意：对于查询条件——月份，此处使用 HQL 的动态参数赋值，使用此方法可避免 SQL 的注入式攻击。

（1）导入 Hibernate 包及 Hibernate 的配置文件 hibernate.cfg.xml。

（2）创建商品销售明细持久化类 SellMonth 及相对应的映射文件 SellMonth.hbm.xml。

（3）创建 SellMonthDao 类，用于封装对数据库的操作。在此类中编写 findMaxSell()方法，用于查询月销售额完成最多的员工。其关键代码如下：

```
public List<SellMonth> findMaxSell(String month){
        List<SellMonth> list = null;
```

```
    try {
        session = HibernateUtils.getSession();                    //获取 Session
        tx = session.beginTransaction();                          //开启事务
        //查询月销售额完成最多的员工
        String hql = "from SellMonth s where s.month = ? and s.totalPrice = ";
        hql += " (select max(totalPrice) from SellMonth s where s.month = ?)";
        Query query = session.createQuery(hql)
                                        .setParameter(0, month)   //为参数赋值
                                        .setParameter(1, month);  //为参数赋值
        list = query.list();
        tx.commit();                                              //提交事务
    } catch (Exception e) {
        e.printStackTrace();                                      //打印异常信息
    }finally{
        HibernateUtils.closeSession(session);                     //关闭 Session
    }
    return list;
}
```

（4）创建名为 SellServlet 的 Servlet 类，用于对业务逻辑进行控制。在此类中通过 doPost()方法对请求进行处理。由于月份销售冠军可能出现并列多人的情况，实例中通过循环获取月份销售冠军。关键代码如下：

```
String command = request.getParameter("command");                //请求类型
SellDao dao = new SellDao();                                     //创建 SellDao 对象
List<SellMonth> list = null;
if("findAll".equals(command)){
    list = dao.findAllSell();                                    //查询所有员工的销售额
}else if("findMax".equals(command)){                             //查询销售额最多的员工
    String month = request.getParameter("month");                //获取月份
    StringBuffer info = new StringBuffer();                      //创建 StringBuffer 对象
    list = dao.findAllSell();                                    //查询所有员工的销量
    List<SellMonth> maxSell = dao.findMaxSell(month);            //查询月份销量最多的员工
    if(maxSell != null && maxSell.size() > 0){
        //通过循环获取月份销售冠军
        for (int i = 0; i < maxSell.size(); i++) {
            SellMonth s = (SellMonth)maxSell.get(i);             //获取结果集中一条记录
            info.append(month + "月份  销售冠军为：" + s.getOperator());
            info.append("销售总额为：" + s.getTotalPrice());
            info.append("<br>");
        }
    }
    //将月销售冠军转换为字符串放入到 request 中
    request.setAttribute("info", info.toString());
}
```

秘笈心法

心法领悟 433：EntityNameResolver 接口。

该接口用于解析给定实体实例的实体名称。一般来说，该接口在动态模型中最为有用。这个接口定义了一个方法 resolveEntityName()，它传递实体实例并预期返回合适的实体名称。

实例 434　限制返回结果的范围

光盘位置：光盘\MR\16\434

高级

实用指数：★★★

在对数据库进行查询操作时，如果一条查询语句返回大量结果集，将严重影响数据库的性能，不仅浪费大量时间，而且由于数据量的问题也给查看结果集带来了诸多不便。在 Hibernate 中，解决这种问题非常简单，它提供了限定返回结果集的范围的方法（此种方法大多用于分页技术中）。

本实例将对学生成绩进行排序，通过对结果集范围的设置，返回前 N 名学生成绩的排名情况，如图 16.20

所示。

学生成绩信息

成绩排名：查询前N名的学生 查询 查看全部学生信息

姓名	年龄	性别	分数	班级	描述
小王	20	男	100	一年一班	学习成绩优秀
小李	21	女	95	一年二班	学习成绩一般
小张	25	女	81.5	一年二班	爱好篮球
小明	22	男	90	一年一班	学习成绩一般
小红	20	女	92	一年二班	爱学习
小井	21	男	70	一年二班	不爱学习
小刘	20	女	99	一年一班	很聪明

图 16.20　学生成绩信息

关键技术

Criteria 接口的 setMaxResults()方法用于设置返回结果的范围，其入口参数为 int 型值。语法如下：

```
public Criteria setMaxResults(int maxResults)
```

参数说明

maxResults：返回结果集的范围。

设计过程

（1）导入 Hibernate 包及 Hibernate 的配置文件 hibernate.cfg.xml。

（2）编写学生持久化类 Student 及相对应的映射文件 Student.hbm.xml。

（3）创建名为 StudentDao 的类（与学生成绩数据相关的数据库操作类），在此类中编写 findAllOrderByGrade()方法，用于获取学习成绩在前 N 名的学生信息。其关键代码如下：

```
public List<Student> findAllOrderByGrade(int i){
	List<Student> list = null;                                  //结果集
	try {
		session = HibernateUtils.getSession();              //获取 Session
		tx = session.beginTransaction();                    //开启事务
		if(i > 0){
			list = session.createCriteria(Student.class)        //创建 Criteria 对象
					.addOrder(Order.desc("grade"))  //设置排序方式
					.setMaxResults(i)               //设置结果集范围
					.list();                        //获取结果集
		}
		tx.commit();                                        //提交事务
	} catch (Exception e) {
		e.printStackTrace();                                //打印异常信息
	}finally{
		HibernateUtils.closeSession(session);               //关闭 Session
	}
	return list;
}
```

此方法的入口参数为 int 型数值，用于设置结果集的范围。在查询的过程中，添加了按成绩升序排序的方式，从而使返回的结果集是学习成绩在前 N 名的学生信息。

（4）创建名为 StudentServlet 的 Servlet 类，在此类中通过 command 参数判断请求类型。当 command 的值为 find 时，查询所有学生信息；当 command 参数为 findAllOrderByGrade 时，查询成绩在指定排名内的学生。其关键代码如下：

```
String command = request.getParameter("command");          //请求类型
StudentDao dao = new StudentDao();                         //实例化 StudentDao 对象
```

```
List<Student> list = null;                                      //结果集
if("find".equals(command)){                                     //按成绩排名查询所有学生
        list = dao.findAllStudents();                           //查询全部学生信息
}else if("findAllOrderByGrade".equals(command)){                //按成绩排名查询所有学生
 String id = request.getParameter("id");                        //获取结果信息的范围
        //查询学习成绩在前 N 名的学生
        list = dao.findAllOrderByGrade(Integer.parseInt(id));
}
request.setAttribute("list", list);                             //将结果集放到 request 中
//转发到 student_list.jsp 页面
request.getRequestDispatcher("student_list.jsp").forward(request, response);
```

当查询成绩在指定排名内的学生信息时，通过 request 获取 JSP 页面传入的排序（升序或降序）排名范围，调用 StudentDao 的 findAllOrderByGrade()方法进行获取。

（5）创建用于显示学生成绩信息的页面 student_list.jsp。

秘笈心法

心法领悟 434：使用 Servlet 容器的注意事项。

由于 Servlet 容器与 Servlet 程序是按 Servlet 接口中约定的方法进行通信的，对于 Servlet 程序中的多个重载形式的 init()方法，Servlet 容器始终只会调用 Servlet 接口中定义的 init()方法，而其他形式的 init()方法则相当于 Servlet 程序内部定义的普通方法，Servlet 容器根本就不知道它们的存在。

实例 435　分页查询数据

光盘位置：光盘\MR\16\435

中级　实用指数：★★☆

实例说明

本实例实现的是分页查询员工信息表中的数据。运行实例，即可看到一共分为 4 页，每个页面显示 3 条员工信息。在文本框中输入需要跳转的页码，单击“确定”按钮，即可跳转到该页码对应的页面。本实例的运行结果如图 16.21 所示。

Hibernate实现数据分页显示

当前是第 2/4 页 第 [ ] 页 确定　上一页　下一页

序号	姓名	部门	性别	出生日期	身份证号	学历	民族	籍贯	家庭住址
1	D	测试部	男	1973-01-01	09231458	本科	汉族	长春市	宽城区
2	E	测试部	男	1980-12-04	93123940	专科	汉族	长春市	朝阳区
3	F	测试部	女	1932-10-10	09345672	专科	汉族	长春市	二道区

图 16.21　分页查询数据

对于数据的分页显示，可以通过 Hibernate 提供的分批检索功能来实现。在本实例中开发了一个可重用的分页 Bean（mrmwq.PaginationInfo.java），在该分页 Bean 中定义了许多属性用于保存相关的分页信息。同时本实例涉及了 setFirstResult()方法和 setMaxResults()方法，它们都是 Query 类中的方法。下面对这两个方法进行详细的介绍。

（1）setFirstResult()方法

该方法用于设置在数据库中查询结果集的起始范围，它可以被 Query 类的实例对象调用。语法如下：

```
setFirstResult(int num)
```

参数说明

num：查询范围的起始记录数。

例如，要从数据库的第 5 条数据开始查询，应该执行如下代码：

```
SetFirstResult(5)
```

（2）setMaxResults()方法

该方法用于设置在数据库中查询结果集的大小，也就是说，它是对查询数量的约束。语法如下：

```
setMaxResults(int size)
```

参数说明

size：查询结果集的大小约束。

创建可重用的 Bean，在该 Bean 中分别定义了多个方法，每个方法都是为了获取不同的信息。

（1）初始化分页信息的方法，具体代码如下：

```
public void init(Session session, String hql, int pageCount) {                    //初始化分页信息
      this.session = session;
      this.hql = hql;
      this.pageCount = pageCount;                                                  //定义每页显示记录数
      totalCount = ((Long) session.createQuery("select count(*)" + hql).
                  uniqueResult()).intValue();                                      //初始化总记录数
      if (this.totalCount % this.pageCount == 0) {                                 //计算需要的总页数
            this.totalPageNum = this.totalCount / this.pageCount;
      } else {
            this.totalPageNum = this.totalCount / this.pageCount + 1;
      }
      if (this.totalPageNum > 1) {                                                 //判断是否有下一页
            this.hasNextPage = true;
      }
}
```

（2）获得前一页信息的方法，具体代码如下：

```
public List getPreviousPage(Session session) {                                     //获得前一页信息
            this.session = session;
            if (this.getCurrentPageNum() > 1) {                                    //有上一页的前提条件是当前页不是第一页
                  this.currentPageNum = this.currentPageNum - 1;
                  if (this.currentPageNum > 1) {
                        this.hasPreviousPage = true;
                        this.hasNextPage = true;
                  } else {
                        this.hasPreviousPage = false;
                        this.hasNextPage = true;
                  }
            }
            return this.getPageMessage();                                          //返回页面信息
      }
```

（3）获取当前页信息的方法，具体代码如下：

```
public List getCurrentPage(Session session) {                                      //获得当前页信息
            this.session = session;
            return this.getPageMessage();
      }
```

（4）获取下一页信息的方法，具体代码如下：

```
public List getNextPage(Session session) {                                         //获得下一页信息
            this.session = session;
            if (this.getCurrentPageNum() < this.totalPageNum) {                    //有下一页的前提条件是当前页不是最后一页
                  this.currentPageNum = this.currentPageNum + 1;
                  if (this.currentPageNum < this.totalPageNum) {                   //计算是否有上一页和下一页
                        this.hasPreviousPage = true;
                        this.hasNextPage = true;
                  } else {
                        this.hasPreviousPage = true;
                        this.hasNextPage = false;
                  }
            }
            return this.getPageMessage();                                          //返回页面信息
      }
```

（5）获取指定页信息的方法，具体代码如下：

```
public List getAppointPage(Session session, int currentPageNum) {                  //获得指定页信息
            this.session = session;
```

```
            if (currentPageNum > 0 & currentPageNum <= this.totalPageNum) {          //必须在有效的页面范围内
                this.currentPageNum = currentPageNum;
                if (this.currentPageNum > 1) {                                        //计算是否有上一页
                    this.hasPreviousPage = true;
                } else {
                    this.hasPreviousPage = false;
                }
                if (this.currentPageNum < this.totalPageNum) {                        //计算是否有下一页
                    this.hasNextPage = true;
                } else {
                    this.hasNextPage = false;
                }
            }
            return this.getPageMessage();                                             //返回页面信息
        }
```

（6）检索指定信息的方法，具体代码如下：

```
private List getPageMessage() {                                                       //检索指定信息的方法
            Query q = session.createQuery(hql);
            q.setFirstResult((this.currentPageNum - 1) * this.pageCount);             //设定起始范围
            q.setMaxResults(this.pageCount);                                          //设定查询结果集的大小
            this.print();
            return q.list();
        }
```

秘笈心法

心法领悟 435：初始化分页信息。

在初始化分页信息 init()方法中需要通过传入的 HQL 语句统计总记录数，所以在 HQL 语句中不能包含 order by 等聚集函数，即 init()方法的第 2 个入口参数的格式必须为“from * where *”，后面不能跟随 order by 等其他子句；如果需要，读者可以在此基础上开发。

实例 436　利用 QBC 检索字段为空的记录

光盘位置：光盘\MR\16\436

中级

实用指数：★★☆

实例说明

本实例实现的是检索图书存放位置为空的记录。运行结果如图 16.22 所示。

QBC检索字段为空的记录

图书编号	图书名称	图书作者	图书价钱	出版地点	存放位置
2	C语言程序设计	IT精英	37	吉林	null
3	SQL应用程序范例宝典	明日科技	79	长春	null
4	Hibernate应用开发完全手册	明日科技	59	长春	null
6	Java程序设计标准教程	明日科技	59	长春	null

图 16.22　利用 QBC 检索字段为空的记录

关键技术

在 Hibernate 应用中使用 QBC（Query By Criteria）查询通常要经过如下 3 个步骤。

（1）使用 Session 实例的 createCriteria()方法创建 Criteria 对象。

（2）使用工具类 Restrictions 的相关方法为 Criteria 对象设置查询条件。该工具类中的一些常用方法如下。

- Restrictions.eq：等于。
- Restrictions.allEq：使用 Map、key/value 进行多个等于的比对。
- Restrictions.gt：大于。

- Restrictions.ge：大于等于。
- Restrictions.ct：小于。
- Restrictions.le：小于等于。
- Restrictions.between：对应 SQL 的 BETWEEN 子句。
- Restrictions.like：对应 SQL 的 like 子句。
- Restrictions.in：对应 SQL 的 in 子句。

（3）使用 Criteria 对象的 list()方法执行查询，同时返回查询结果。

设计过程

（1）创建用于操作数据库的 DAO 类，在该类中用 QBC 检索方式查询出图书存放位置为空的记录，同时返回一个 list 集合。具体代码如下：

```
public List QueryBook(){
        Session session = null;
        List list=new ArrayList();
      try {
            session = HibernateUtil.getSession();                          //得到 Session 对象
            session.beginTransaction();                                     //开启事务
         Criteria criteria=session.createCriteria(Book.class);
          criteria.add(Restrictions.isNull("store"));                    //查询 store 字段为空的记录
          list=criteria.list();
      } catch (Exception e) {
            e.printStackTrace();
            session.getTransaction().rollback();                           //事务回滚
      }finally{
            HibernateUtil.closeSession(session);                          //关闭 Session
      }
      return list;                                                          //返回 list 集合
  }
```

（2）创建用于显示数据库记录的 index.jsp 页面，具体代码如下：

```
<c:forEach items="${requestScope.list}" var="st" >                      //JSTL 表达式遍历 list 集合中数据
      <tr>                                                                //循环输出 list 集合中数据
         <td height="27"><div align="center">${st.id}</div></td>
         <td><div align="center">${st.name}</div></td>
         <td><div align="center">${st.author}</div></td>
         <td><div align="center">${st.price}</div></td>
         <td><div align="center">${st.place}</div></td>
         <td><div align="center">
         <c:if test="${empty st.store}">
            <c:out value="null"></c:out>
         </c:if>
         </div></td>
      </tr>
   </c:forEach>
```

秘笈心法

心法领悟 436：QBC 查询语句的优点。

QBC（Query By Criteria）是 Hibernate 提供的一种“更加面向对象”检索方式，在条件查询方面比 HQL 更加灵活，同时支持在运行时动态地生成查询语句。

实例 437	利用 QBC 检索不满足指定条件的记录 光盘位置：光盘\MR\16\437	中级 实用指数：★★☆

实例说明

本实例实现的是在学生信息表中查询成绩不在 60～80 分之间的学生信息。运行本实例，即可看到学生信息

表中的全部数据。单击“查询不满足条件的记录”按钮，即可将学生信息表中成绩不在 60～80 分之间的学生信息显示在下面的表格中，如图 16.23 所示。

查询不满足条件的记录>>>

成绩不在60分－80分之间的学生信息　[查询不满足条件的记录]

编号	姓名	性别	年龄	班级	分数
1	张远	男	19	20801班	89分
4	刘马媛	女	21	20803班	84分
5	甄忠利	男	23	20803班	90分
6	杨明月	女	24	20804班	87分
7	刘晓莉	女	19	20804班	99分

图 16.23　利用 QBC 检索不满足指定条件的记录

关键技术

要实现利用 QBC 检索不满足指定条件的记录，就要用到 Restrictions 工具类。在该类中定义了多种方法为 Criteria 对象设置查询条件，如在本实例中就用到了 Restrictions 工具类的 not()方法和 between()方法，其中的 not()方法用于查询不满足条件的记录，between()方法用于查询指定区间的记录。

（1）创建用于操作数据库的 DAO 类，在该类中使用 QBC 检索方式查询出学生信息表中全部数据和不在指定区间内的数据，同时这两个方法都返回一个 list 集合。具体代码如下：

```
public List QueryStu(){
        Session session = null;
        List list=new ArrayList();
        try {
                session = HibernateUtil.getSession();                                    //得到 Session 对象
                session.beginTransaction();                                              //开启事务
            Criteria c=session.createCriteria(Student.class);
            c.add(Restrictions.not(Restrictions.between("grade", new Integer(60), new Integer(80))));  //查询不满足条件的记录
            list=c.list();
        } catch (Exception e) {
                e.printStackTrace();
                session.getTransaction().rollback();                                     //事务回滚
        }finally{
                HibernateUtil.closeSession(session);                                     //关闭 Session
        }
        return list;                                                                     //返回 list 集合
 }
public List QueryAll(){
         Session session = null;
         List list=new ArrayList();
        try {
                session = HibernateUtil.getSession();                                    //得到 Session 对象
                session.beginTransaction();                                              //开启事务
            Criteria c=session.createCriteria(Student.class);
            list=c.list();                                                               //查询所有记录
        } catch (Exception e) {
                e.printStackTrace();
                session.getTransaction().rollback();                                     //事务回滚
        }finally{
                HibernateUtil.closeSession(session);                                     //关闭 Session
        }
        return list;
 }
```

（2）创建用于显示数据库记录的 index.jsp 页面，具体代码如下：

```
 <c:if test="${empty param.order}">                                                      //判断输入文本框中的数据是否为空
      <c:forEach items="${requestScope.list1}" var="st" >                                //为空时，显示所有数据库记录
         <tr>
             <td height="27"><div align="center">${st.id}</div></td>
             <td><div align="center">${st.name}</div></td>
             <td><div align="center">${st.sex}</div></td>
             <td><div align="center">${st.age}</div></td>
             <td><div align="center">${st.classname}</div></td>
             <td><div align="center">${st.grade}分</div></td>
         </tr>
      </c:forEach>
```

```
 </c:if>
 <c:if test="${!empty param.order}">                                    //判断输入文本框中的数据是否为空
     <c:forEach items="${requestScope.list}" var="st" >                 //不为空时，会根据输入值查询记录
       <tr>
           <td height="27"><div align="center">${st.id}</div></td>
           <td><div align="center">${st.name}</div></td>
           <td><div align="center">${st.sex}</div></td>
           <td><div align="center">${st.age}</div></td>
           <td><div align="center">${st.classname}</div></td>
           <td><div align="center">${st.grade}分</div></td>
       </tr>
     </c:forEach>
   </c:if>
```

秘笈心法

心法领悟 437：设置文本框隐藏属性。

单击“查询不满足条件的记录”按钮，即可在表格中显示不重复记录的数据。其核心技术是设置了文本框隐藏属性，通过判断文本框中数据是否为空来编写相应的 SQL 语句。文本框隐藏有两种方式，第 1 种是设置 input 标记中 style 属性值为 display:none，第 2 种方式是设置 style 属性值为 visibility:hidden。

实例 438 QBC 忽略大小写查询

光盘位置：光盘\MR\16\438

中级

实用指数：★★☆

实例说明

本实例实现的是查询企业名称时忽略大小写。运行实例，即可在页面中看到全部的企业信息。在文本框中输入数据，如 mr，单击“查询”按钮，即可将企业名称中无论 mr 是大写或是小写的数据检索出来。本实例的运行结果如图 16.24 所示。

QBC忽略大小写查询>>>

请输入企业名称： 查询

企业编号	企业名称	企业地点	企业规模	企业性质	企业属性
1	MR	长春市	200人	私营	计算机
2	mr	吉林市	200人	私营	计算机
3	Mr	四平市	200人	私营	计算机
4	mR	白山市	200人	私营	计算机

图 16.24　QBC 忽略大小写查询

关键技术

要实现 QBC 忽略大小写查询，就要用到 Restrictions 工具类。在该类中定义了多种方法为 Criteria 对象设置查询条件，如在本实例中就用到了 Restrictions 工具类的 eq()方法和 ignoreCase()方法，其中的 eq()方法用于比较是否相等，ignoreCase()方法则用于忽略大小写进行比较。

（1）创建用于操作数据库的 DAO 类，在该类中用 QBC 检索方式查询出企业信息表中的全部数据和字段是忽略大小写的数据，同时这两个方法都返回一个 list 集合。具体代码如下：

```
public List QueryDep(String name){
        Session session = null;
        List list=new ArrayList();
        try {
                session = HibernateUtil.getSession();                   //得到 Session
                session.beginTransaction();                             //开启事务
            Criteria c=session.createCriteria(Depart.class);
            c.add(Restrictions.eq("name", name).ignoreCase());          //检索指定字段时忽略大小写
            list=c.list();
        } catch (Exception e) {
                e.printStackTrace();
                session.getTransaction().rollback();                    //事务回滚
        }finally{
                HibernateUtil.closeSession(session);                    //关闭 Session
```

```
        }
        return list;                                              //返回 list 集合
 }
public List QueryAll(){
          Session session = null;
          List list=new ArrayList();
        try {
                session = HibernateUtil.getSession();             //得到 Session
                session.beginTransaction();                       //开启事务
           Criteria c=session.createCriteria(Depart.class);
           list=c.list();                                         //检索全部数据
        } catch (Exception e) {
                e.printStackTrace();
                session.getTransaction().rollback();              //事务回滚
        }finally{
                HibernateUtil.closeSession(session);              //关闭 Session
        }
        return list;                                              //返回 list 集合
 }
```

（2）创建用于显示数据库记录的 index.jsp 页面，具体代码如下：

```
<c:if test="${empty param.name}">                                 //判断输入文本框中的数据是否为空
       <c:forEach items="${requestScope.list1}" var="st" >        //为空时，显示所有数据库记录
          <tr>
              <td height="27"><div align="center">${st.id}</div></td>
              <td><div align="center">${st.name}</div></td>
              <td><div align="center">${st.address}</div></td>
              <td><div align="center">${st.scope}</div></td>
              <td><div align="center">${st.quality}</div></td>
              <td><div align="center">${st.trace}</div></td>
          </tr>
        </c:forEach>
  </c:if>
  <c:if test="${!empty param.name}">                              //判断输入文本框中的数据是否为空
       <c:forEach items="${requestScope.list}" var="st" >         //不为空时，会根据输入值查询记录
           <tr>
              <td height="27"><div align="center">${st.id}</div></td>
              <td><div align="center">${st.name}</div></td>
              <td><div align="center">${st.address}</div></td>
              <td><div align="center">${st.scope}</div></td>
              <td><div align="center">${st.quality}</div></td>
              <td><div align="center">${st.trace}</div></td>
          </tr>
       </c:forEach>
     </c:if>
```

秘笈心法

心法领悟 438：${empty param.name}的使用。

这是 EL 表达式的写法，所有的 EL 都是以“${”为起始，以“}”为结尾的。其中 param 是与输入有关的 EL 表达式隐含对象（还有一个是 paramValues）。一般来说，在 JSP 中获取用户请求参数时，可以利用 request.getParameter(String name)，但是在 EL 表达式中可以直接用${param.name}获取。

实例 439	利用 QBC 查询满足指定范围的所有记录 光盘位置：光盘\MR\16\439	中级 实用指数：★★☆

实例说明

本实例实现的是在学生信息表中查询成绩在 80～90 分之间的学生信息。运行实例，即可看到学生信息表中的全部数据。单击“查询满足条件的记录”按钮，即可将学生信息表中成绩在 80～90 分之间的学生信息显示在下面的表格中，如图 16.25 所示。

查询满足条件的记录>>>

成绩在80分一90分之间的学生信息　查询满足条件的记录

编号	姓名	性别	年龄	班级	分数
1	张远	男	19	20801班	89分
4	刘马媛	女	21	20803班	84分
5	甄忠利	男	23	20803班	90分
6	杨明月	女	24	20804班	87分

图 16.25　利用 QBC 查询满足指定范围的所有记录

关键技术

要实现利用 QBC 检索满足指定条件的记录，就要使用到 Restrictions 工具类。在该类中定义了多种方法为 Criteria 对象设置查询条件。例如，在本实例中就用到了 Restrictions 工具类的 between()方法，该方法主要用于查询指定区间的记录。

（1）创建用于操作数据库的 DAO 类，在该类中用 QBC 检索方式查询出学生信息表中的全部数据和在指定区间内的数据，同时这两个方法都返回一个 list 集合。具体代码如下：

```
public List QueryStu(){
        Session session = null;
        List list=new ArrayList();
        try {
            session = HibernateUtil.getSession();                          //得到 Session 对象
            session.beginTransaction();                                    //开启事务
          Criteria c=session.createCriteria(Student.class);
          c.add(Restrictions.between("grade", new Integer(80), new Integer(90)));   //查询满足条件记录
          list=c.list();
        } catch (Exception e) {
            e.printStackTrace();
            session.getTransaction().rollback();                           //事务回滚
        }finally{
            HibernateUtil.closeSession(session);                           //关闭 Session
        }
        return list;                                                       //返回 list 集合
 }
public List QueryAll(){
         Session session = null;
         List list=new ArrayList();
        try {
            session = HibernateUtil.getSession();                          //得到 Session 对象
            session.beginTransaction();                                    //开启事务
          Criteria c=session.createCriteria(Student.class);
          list=c.list();                                                   //查询所有记录
        } catch (Exception e) {
            e.printStackTrace();
            session.getTransaction().rollback();                           //事务回滚
        }finally{
            HibernateUtil.closeSession(session);                           //关闭 Session
        }
        return list;
 }
```

（2）创建用于显示数据库记录的 index.jsp 页面，具体代码如下：

```
<c:if test="${empty param.order}">                                         //判断输入文本框中的数据是否为空
      <c:forEach items="${requestScope.list1}" var="st" >                  //为空时，显示所有数据库记录
        <tr>
            <td height="27"><div align="center">${st.id}</div></td>
            <td><div align="center">${st.name}</div></td>
            <td><div align="center">${st.sex}</div></td>
            <td><div align="center">${st.age}</div></td>
            <td><div align="center">${st.classname}</div></td>
            <td><div align="center">${st.grade}分</div></td>
        </tr>
```

```
        </c:forEach>
    </c:if>
    <c:if test="${!empty param.order}">                                    //判断输入文本框中的数据是否为空
        <c:forEach items="${requestScope.list}" var="st" >                 //不为空时，会根据输入值查询记录
          <tr>
              <td height="27"><div align="center">${st.id}</div></td>
              <td><div align="center">${st.name}</div></td>
              <td><div align="center">${st.sex}</div></td>
              <td><div align="center">${st.age}</div></td>
              <td><div align="center">${st.classname}</div></td>
              <td><div align="center">${st.grade}分</div></td>
          </tr>
        </c:forEach>
      </c:if>
```

秘笈心法

心法领悟 439：Restrictions.between()方法的作用范围。

Restrictions.between()方法对应于 SQL 的 BETWEEN 子句，BETWEEN 运算符是具有包含性的，即首尾的值也是符合条件的。



实例 440　利用 HQL 实现模糊查询　中级

光盘位置：光盘\MR\16\440　实用指数：★★☆

实例说明

本实例实现的是利用 HQL 模糊查询员工基本信息。运行实例，输入用户名的姓，如“李”，单击“查询”按钮，即可查询出所有姓“李”的员工并显示在表格中，如图 16.26 所示。

请输入用户名：　查询

HQL实现模糊查询

编号	姓名	性别	年龄	部门
3	李丽丽	女	19	C部门
4	李阳	男	27	Asp部门
5	李乾坤	男	21	C部门
6	李烨	男	37	PHP部门

图 16.26　利用 HQL 实现模糊查询

关键技术

要用 HQL 实现模糊查询员工信息，就要在 HQL 语句中使用通配符“%”和 like 谓词，下面分别介绍。

（1）%通配符

“%”是通配符，可匹配 0 个或更多任意长度的字符串。在 SQL 语言中，最常用的通配符可能就是“%”。可以在查询条件的任意位置放置一个“%”来代表一个任意长度的字符串，也可以放置两个“%”进行查询，但是最好不要连续出现两个“%”。

注意：在 Microsoft Access 数据库中不能使用“%”，其功能由“*”通配符所替代。

（2）like 谓词

like 谓词用于查找字符串；使用时，取“%”代表任意字符串。其具体形式如下（以 mr 字符为例）。

- 包含字符 mr 的任意文本：like'%mr%'。
- 以字符 mr 开头的任意文本：like'mr%'。
- 以字符 mr 结尾的任意文本：like'%mr'。

（1）创建用于操作数据库的 DAO 类，在该类中根据输入文本框中的字符串用 HQL 检索方式模糊查询出学生信息表中的数据，同时该方法返回一个 list 集合。具体代码如下：

```
public List QueryPer(String name){
            Session session = null;
            List li=new ArrayList();
            try {
```

```
            session = HibernateUtil.getSession();                         //得到 Session 对象
            session.beginTransaction();                                   //开启事务
            Query query=session.createQuery("from Person where name like '%"+name+"%'");   //HQL 进行模糊查询
            li=query.list();
        } catch (Exception e) {
                e.printStackTrace();
                session.getTransaction().rollback();                      //事务回滚
        }finally{
                HibernateUtil.closeSession(session);                      //关闭 Session
    }
        return li;                                                        //返回 list 集合
    }
```

（2）创建用于根据查询条件显示数据库记录的 index.jsp 页面，具体代码如下：

```
<c:forEach items="${requestScope.list1}" var="per" >                      //遍历 list 集合
        <tr>
          <td height="27"><div align="center">${per.id}</div></td>        //循环输出 list 集合中数据
          <td><div align="center">${per.name}</div></td>
          <td><div align="center">${per.sex}</div></td>
          <td><div align="center">${per.age}</div></td>
          <td><div align="center">${per.depart}</div></td>
        </tr>
</c:forEach>
```

秘笈心法

心法领悟 440：like 谓词的作用范围。

like 谓词只用于下列数据类型：char、nchar、varchar、nvarchar 和 datetime。

实例 441　利用 QBC 实现模糊查询

光盘位置：光盘\MR\16\441

中级

实用指数：★★☆

实例说明

本实例实现的是利用 QBC 模糊查询员工基本信息。运行实例，输入用户名的姓，如“王”，单击“查询”按钮，即可查询出所有姓“王”的员工并显示在表格中，如图 16.27 所示。

请输入用户名：［　　］ 查询

QBC实现模糊查询

编号	姓名	性别	年龄	部门
7	王明远	男	17	Asp部门
8	王琳	男	30	JavaWeb部门

图 16.27　利用 QBC 实现模糊查询

关键技术

要实现 QBC 模糊查询，就要用到 Restrictions 工具类。在该类中定义了多种方法为 Criteria 对象设置查询条件。例如，在本实例中就用到了 Restrictions 工具类的 like()方法，该方法对应 SQL 的 like 子句。

（1）创建用于操作数据库的 DAO 类，在该类中根据输入文本框中的字符串用 HQL 检索方式模糊查询出学生信息表中的数据，同时该方法返回一个 list 集合。具体代码如下：

```
public List QueryPer(String name){
        Session session = null;
        List li=new ArrayList();
        try {
            session = HibernateUtil.getSession();                         //得到 Session 对象
            session.beginTransaction();                                   //开启事务
            Criteria criteria=session.createCriteria(Person.class);
            criteria.add(Restrictions.like("name","%"+name+"%"));         //QBC 模糊查询
             li=criteria.list();
        } catch (Exception e) {
              e.printStackTrace();
              session.getTransaction().rollback();                        //事务回滚
```

```
        }finally{
            HibernateUtil.closeSession(session);                    //关闭 Session
        }
        return li;                                                  //返回 list 集合
    }
```

（2）创建用于根据查询条件显示数据库记录的 index.jsp 页面，具体代码如下：

```
<c:forEach items="${requestScope.list1}" var="per" >                 //遍历 list 集合
    <tr>
      <td height="27"><div align="center">${per.id}</div></td>       //循环输出 list 集合中数据
      <td><div align="center">${per.name}</div></td>
      <td><div align="center">${per.sex}</div></td>
      <td><div align="center">${per.age}</div></td>
      <td><div align="center">${per.depart}</div></td>
    </tr>
 </c:forEach>
```

秘笈心法

心法领悟 441：Restrictions.like()方法中参数的使用。

该方法的第 1 个参数是对象中属性的名称，而第 2 个则是 value 值。本实例中该方法的第 2 个参数 name 值是一个变量，所以两边都要用“+”进行连接。

实例 442　HQL 在查询中使用统计函数

光盘位置：光盘\MR\16\442

中级

实用指数：★★☆

实例说明

本实例实现的是 HQL 在查询中使用函数统计学生总成绩。运行实例，即可看到学生信息表中的全部信息，同时在表格最下方对学生的总成绩进行了汇总，如图 16.28 所示。

HQL中使用统计函数

编号	姓名	性别	年龄	班级	分数
1	张远	男	19	20801班	89分
2	刘丽	女	18	20801班	72分
3	杨林	男	22	20802班	71分
4	刘马媛	女	21	20803班	84分
5	甄忠利	男	23	20803班	90分
6	杨明月	女	24	20804班	87分
7	刘晓莉	女	19	20804班	99分
合计：					592分

图 16.28　HQL 在查询中使用统计函数

关键技术

SUM 聚集函数主要用于返回表达式中所有值的和，或只返回 DISTINCT 值。该函数只能用于数据类型是数字的列，null 值将被忽略。

语法：

```
SUM ( [ALL|DISTINCT] expression )
```

参数说明

❶ ALL：对所有的值进行聚集函数运算。ALL 是默认设置。

❷ DISTINCT：指定 SUM 返回唯一值的和。

❸ expression：是常量、列或函数，或者是算术、按位与字符串等运算符的任意组合。expression 是精确数字或近似数字数据类型（bit 数据类型除外）的表达式，不允许使用聚集函数和子查询。

（1）创建用于操作数据库的 DAO 类，在该类中查询出学生信息表中的全部数据并统计成绩，同时这两个方法都返回一个 list 集合。具体代码如下：

```
public class StudentDAO {
    public long QuerySum(){
        Session session = null;
        long i = 0;
        try {
            session = HibernateUtil.getSession();                       //得到 Session 对象
            session.beginTransaction();                                 //开启事务
```

```
            i=(Long) session.createQuery("select sum(grade) from Student").uniqueResult();      //对成绩进行汇总
        } catch (Exception e) {
            e.printStackTrace();
            session.getTransaction().rollback();                                                  //事务回滚
        }finally{
            HibernateUtil.closeSession(session);                                                  //关闭 Session
        }
        return i;
    }
public List QueryStu(){
    Session session = null;
    List list=new ArrayList();
    try {
        session = HibernateUtil.getSession();                                                     //得到 Session 对象
        session.beginTransaction();                                                               //开启事务
      list=session.createQuery("from Student").list();                                            //查询所有数据
    } catch (Exception e) {
        e.printStackTrace();
    session.getTransaction().rollback();                                                          //事务回滚
    }finally{
        HibernateUtil.closeSession(session);                                                      //关闭 Session
    }
    return list;
 }
```

（2）创建显示学生信息和成绩汇总的 index.jsp 页面，具体代码如下：

```
<c:forEach items="${requestScope.list}" var="st" >                                                //遍历 list 集合中数据
    <tr>
      <td height="27"><div align="center">${st.id}</div></td>                                      //循环输出 list 集合中数据
      <td><div align="center">${st.name}</div></td>
      <td><div align="center">${st.sex}</div></td>
      <td><div align="center">${st.age}</div></td>
      <td><div align="center">${st.classname}</div></td>
      <td><div align="center">${st.grade}分</div></td>
    </tr>
 </c:forEach>
    <tr>
      <td colspan="5"><strong>  合计:</strong></td>
      <td><div align="center"><c:out value="${requestScope.sum}"/>分</div></td>                  //得到 request 范围内的 sum 值
    </tr>
```

■ 秘笈心法

心法领悟 442：SUM 函数的作用范围。

SUM 函数只能用于数据类型是 int、smallint、tinyint、decimal、numeric、float、real、money 和 smallmoney 的字段。

实例 443　利用 HQL 实现投影查询

光盘位置：光盘\MR\16\443

中级

实用指数：★★☆

■ 实例说明

本实例实现的是利用 HQL 投影查询学生信息表中的部分信息，如图 16.29 所示。

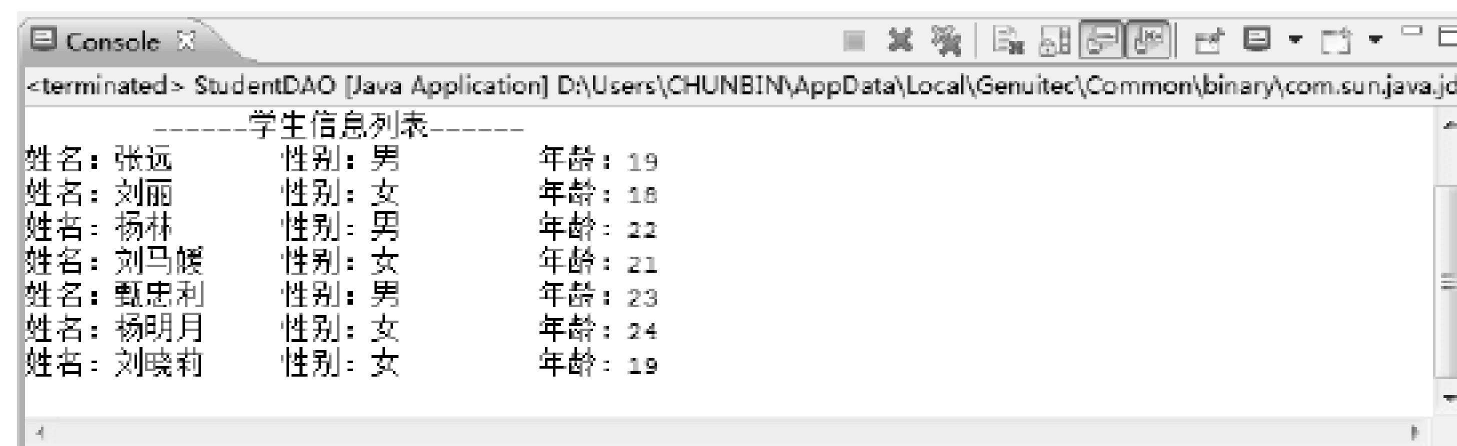


图 16.29　利用 HQL 实现投影查询

■

要实现投影查询学生的部分信息，只需加载需要的那部分属性即可。此时返回的结果是 Object 型数组组成的结果集，可以通过遍历 list 集合得到每一个 Object 类型数组，同时把该数组中的信息显示出来。

在执行投影查询时，也可以通过实例化检索结果，使程序在访问数据时能够运用面向对象的思想。同样检索持久化类 Student 的 name、sex、age 属性，也可以采用本例中的另一种编码方式。代码如下：

```
list=session.createQuery("select new Student(name,sex,age) from Student").list();
```

采用上述编码方式需要一个前提条件，就是必须在对应的持久化类 Student 中定义一个相对应的构造方法，供实例化检索结果时使用。

设计过程

创建用于操作数据库的 DAO 类，在该类中查询出学生的部分信息，并将其显示出来。具体代码如下：

```
public void QueryStu(){
        Session session = null;
        List list=new ArrayList();
        try {
                session = HibernateUtil.getSession();                        //得到 Session 对象
                session.beginTransaction();                                  //开启事务
            list=session.createQuery("select name,sex,age from Student").list();  //查询学生表中部分属性
            System.out.println("\t------学生信息列表------");
            for(int i=0;i<list.size();i++)                                   //遍历 list 集合
            {
                Object[] object=(Object[]) list.get(i);                      //得到每一个 Object 数组
                System.out.print("姓名："+object[0]+"\t");                  //循环输出数组中的属性值
                System.out.print("性别："+object[1]+"\t");
                System.out.println("\t 年龄："+object[2]);
            }
        } catch (Exception e) {
                e.printStackTrace();
                session.getTransaction().rollback();                         //事务回滚
        }finally{
                HibernateUtil.closeSession(session);                         //关闭 Session
        }
 }
```

秘笈心法

心法领悟 443：投影查询的概念。

当只需要访问一个持久化类的部分属性时，可以只加载需要的那部分属性，不过此时的返回值不再是由持久化对象组成的结果集，而是由 Object 型数组组成的结果集，这种只检索部分属性的查询方式就叫做投影查询。

实例 444	QBC 实现将查询结果排序 光盘位置：光盘\MR\16\444	中级 实用指数：★★☆

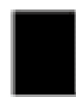
实例说明

本实例实现的是用 QBC 方式对学生成绩进行排序。运行实例，即可看到学生成绩按照升序排序并显示在页面中，如图 16.30 所示。

QBC实现成绩升序排序

编号	姓名	性别	年龄	班级	分数
3	杨林	男	22	20802班	71分
2	刘丽	女	18	20801班	72分
4	刘马媛	女	21	20803班	84分
6	杨明月	女	24	20804班	87分
1	张远	男	19	20801班	89分
5	魏忠利	男	23	20803班	90分
7	刘晓莉	女	19	20804班	99分

图 16.30　QBC 实现将查询结果排序



Hibernate 支持对查询结果进行排序，HQL 与 SQL 的实现方式相同，通过 order by 关键字实现；QBC 则通过 Criteria 类的 addOrder (Order order)方法实现，其入口参数为 org.hibernate.criterion.Order 类的对象。Order 类提供了两个静态方法，分别介绍如下。

- asc(String attribute)：升序排序。
- desc(String attribute)：降序排序。

以上两个静态方法的入口参数均为排序依据的持久化类的属性名称。

设计过程

（1）创建用于操作数据库的 DAO 类，在该类中用 QBC 检索方式对学生成绩进行升序排序，同时返回一个 list 集合。具体代码如下：

```
public List QueryStu(){
        Session session = null;
        List list=new ArrayList();
        try {
                session = HibernateUtil.getSession();                          //得到 Session 对象
                session.beginTransaction();                                    //开启事务
            Criteria c=session.createCriteria(Student.class);
            c.addOrder(Order.asc("grade"));                                    //对学生成绩升序排序
            list=c.list();
        } catch (Exception e) {
                e.printStackTrace();
                session.getTransaction().rollback();                           //事务回滚
        }finally{
                HibernateUtil.closeSession(session);                           //关闭 Session
        }
        return list;                                                           //返回 list 集合
  }
```

（2）创建显示学生升序排序信息的 index.jsp 页面，具体代码如下：

```
<c:forEach items="${requestScope.list}" var="st" >                            //遍历 list 集合中数据
        <tr>
            <td height="27"><div align="center">${st.id}</div></td>            //循环输出 list 集合中数据
            <td><div align="center">${st.name}</div></td>
            <td><div align="center">${st.sex}</div></td>
            <td><div align="center">${st.age}</div></td>
            <td><div align="center">${st.classname}</div></td>
            <td><div align="center">${st.grade}分</div></td>
        </tr>
</c:forEach>
```

秘笈心法

心法领悟 444：多个条件排序查询。

在本实例中若是根据成绩和年龄进行升序排序，则可以通过 c.addOrder(Order.asc("grade"))和 c.addOrder (Order.asc("age"))代码来实现，这两行代码的意思是首先根据成绩进行升序排序，成绩相同的学生再根据年龄进行升序排序。

实例 445　HQL 内连接查询商品信息

光盘位置：光盘\MR\16\445

中级

实用指数：★★✩

实例说明

本实例实现的是内连接查询商品信息。运行实例，即可在表格中看到商品信息表和商品类型表中的数据。在该表格中多个商品可以属于一个商品类型，同时一个商品类型也可以包括多个商品。本实例的运行结果如图 16.31 所示。

HQL内连接查询商品信息

编号	商品名称	商品价格	生产日期	生产地区	商品类型
1	苹果	2元	2010-09-02	河南	水果
2	香蕉	3元	2010-10-01	河南	水果
3	羽绒服	870元	2010-10-09	吉林	服装
4	牛仔裤	200元	2010-11-01	山东	服装
5	电冰箱	2900元	2010-12-01	山东	电器
6	洗衣机	1000元	2010-12-02	河北	电器
7	电视机	3100元	2011-01-02	吉林	电器

图 16.31　HQL 内连接查询商品信息

在 SQL 中通过 JOIN 子句可以实现多表之间的联合查询。与 SQL 查询一样，HQL 也支持各种连接查询。HQL 中提供了以下几种连接查询。

- 内连接（Inner Join）。
- 左外连接（Left Outer Join）。
- 右外连接（Right Outer Join）。
- 全连接（Full Join）。

在本实例中使用的是内连接查询数据，其语法如下：

```
FROM PersistentClassesName alias INNER JOIN FETCH alias.attributeName
```

参数说明

❶ PersistentClassesName：代表主表或从表对应的持久化类。

❷ alias：为 PersistentClassesName 持久化类指定别名。

❸ FETCH：关键字，用于根据主表对象读出从表对象或根据从表对象读出主表对象后立即填充到对应的主表对象或从表对象中。该关键字可以省略，如果省略，即不立即填充到对应的主表或从表对象中，此时得到的结果集的每个记录都是一个 Object 数组。

❹ attributeName：代表主表或从表对应的持久化类中的属性。

设计过程

（1）创建用于操作数据库的 DAO 类，在该类中用 HQL 内连接查询方式查询出商品的基本信息以及商品所属类型，同时返回一个 list 集合。具体代码如下：

```
public List QueryGoods(){
        Session session = null;
        List list=new ArrayList();
        try {
                session = HibernateUtil.getSession();                    //得到 Session 对象
                session.beginTransaction();                              //开启事务
           list=session.createQuery("from Goods g inner join g.type").list();   //查询商品基本信息和所属类型
        } catch (Exception e) {
                e.printStackTrace();
                session.getTransaction().rollback();                     //事务回滚
        }finally{
                HibernateUtil.closeSession(session);                     //关闭 Session
        }
        return list;                                                     //返回 list 集合
 }
```

（2）创建显示商品基本信息和商品所属类型的 index.jsp 页面，具体代码如下：

```
<c:forEach items="${requestScope.list}" var="goods" >                   //循环遍历 list 集合
        <tr>
           <td height="27"><div align="center">${goods[0].id}</div></td>   //输出商品对象中数据
           <td><div align="center">${goods[0].gname}</div></td>
           <td><div align="center">${goods[0].price}元</div></td>
           <td><div align="center">${goods[0].date}</div></td>
           <td><div align="center">${goods[0].place}</div></td>
           <td><div align="center">${goods[1].tname}</div></td>       //输出商品类型对象中数据
        </tr>
</c:forEach>
```

心法领悟 445：内连接的概念。

内连接也称为连接，最早被称为普通连接或自然连接。内连接将从结果中删除其他被连接表中没有匹配行的所有行，所以会丢失信息。在 HQL 检索方式中，内连接通过关键字 INNER JOIN 实现，可以简写为 JOIN。

第 17 章

Hibernate 高级话题

- 关联映射
- Hibernate 检索策略
- Hibernate 集合映射与事务应用

17.1　关 联 映 射

实例 446　关联映射实现级联保存与更新

光盘位置：光盘\MR\17\446

中级

实用指数：★★☆

实例说明

本实例实现的是级联保存和更新员工信息表和部门信息表。运行实例，即可在控制台中看到当保存员工信息时将级联保存员工相对应的部门信息；当更新员工信息表中的部门名称时，将级联更新部门信息表中相对应的部门名称信息。本实例的运行结果如图 17.1 所示。

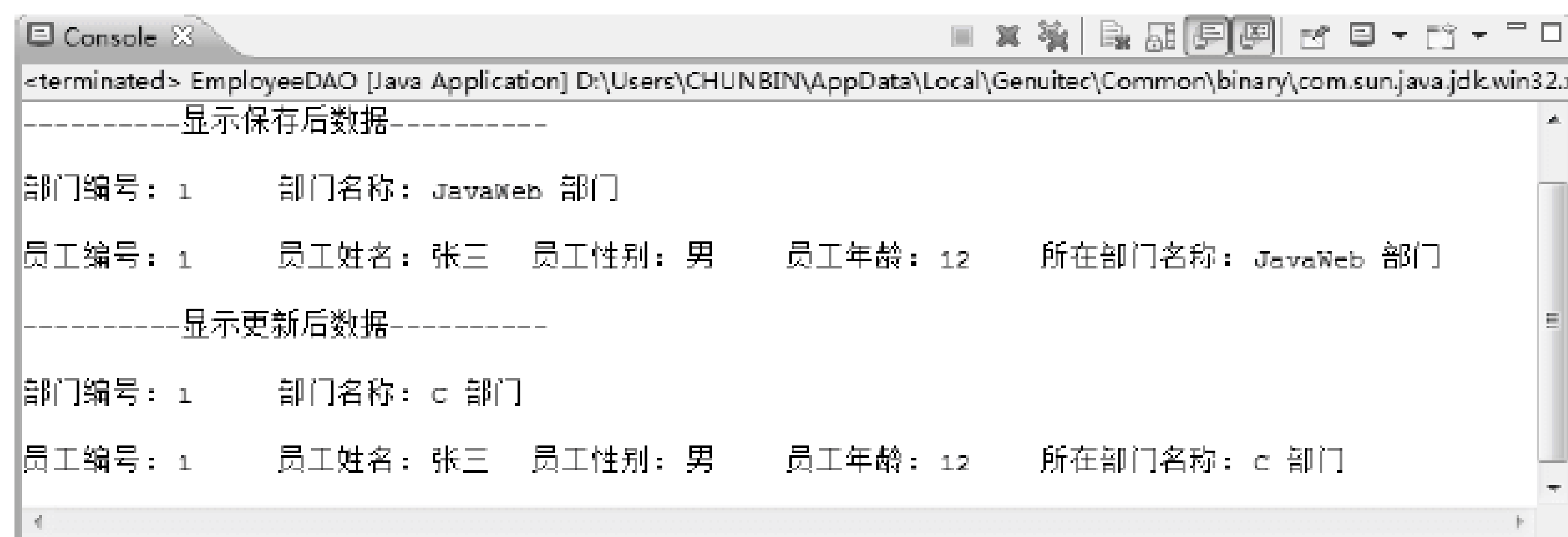


图 17.1　关联映射实现级联保存与更新

关键技术

对数据的级联保存与更新，可以通过设置<many-to-one>元素的 cascade 属性来实现。cascade 的属性值如下。

- none：cascade 属性的默认值。在保存、修改或删除本对象时，不对与之关联的对象进行任何操作。
- save-update：当保存或修改本对象时，级联保存所有与之关联的临时对象，级联更新所有与之关联的游离对象。
- delete：当删除本对象时，级联删除所有与之关联的对象。
- all：包括 save-update 和 delete 的行为。
- delete-orphan：删除所有与本对象解除关联关系的对象。
- all-delete-orphan：包括 all 和 delete-orphan 的行为。

（1）分别创建员工信息类和部门信息类，这两个类主要用于定义员工和部门的一些属性。

（2）配置员工映射文件，具体代码如下：

```
<class name="cn.itcast.mrpojo.Employee" table="tb_employee" >                    //定义表主键的映射
        <id name="id">
            <generator class="native" />
         </id>
         <property name="ename" />                                                //定义属性
         <property name="sex" />
         <property name="age" />
         <many-to-one column="did" name="depart" class="cn.itcast.mrpojo.Depart" cascade="save-update" lazy="false"/>    //定义 many-to-one 元素
</class>
```

（3）配置部门映射文件，具体代码如下：

```
<class name="cn.itcast.mrpojo.Depart" table="tb_depart" >
        <id name="id">                                                    //定义表主键的映射
            <generator class="native" />
         </id>
         <property name="dname" />                                        //定义属性
</class>
```

（4）定义 EmployeDAO 类，该类主要是对数据进行级联保存和更新。

级联保存的具体代码如下：

```
public Boolean SaveEmpl(){
        Boolean flag=true;
                Session session = null;
                try {
                        Employee em=new Employee();
                        Depart depart=new Depart();
                        em.setEname("张三");                                    //设置员工信息
                        em.setAge("12");
                        em.setSex("男");
                        depart.setDname("JavaWeb  部门");
                        em.setDepart(depart);                                   //员工和部门对象映射关联
                        session = HibernateUtil.getSession();
                        session.beginTransaction();
                        session.save(em);                                       //对象保存
                        session.getTransaction().commit();                      //事务提交
                } catch (Exception e) {
                        e.printStackTrace();
                        session.getTransaction().rollback();                    //事务回滚
                        flag=false;
                }finally{
                        HibernateUtil.closeSession(session);                    //关闭 Session
                }
                return flag;
        }
```

级联更新的具体代码如下：

```
public Boolean UpdateEmpl(int did){
        Boolean flag=true;
        EmployeeDAO ed=new EmployeeDAO();
                Session session = null;
                Depart depart=null;
                try {
                        session = HibernateUtil.getSession();
                        session.beginTransaction();
                        Employee em2=(Employee) session.get(Employee.class, new Integer(did));    //查询指定 id 对象
                        Depart de=em2.getDepart();
                        de.setDname("C  部门");                                  //设置新属性
                        session.update(em2);                                    //对象更新
                    session.getTransaction().commit();                          //事务提交
                } catch (Exception e) {
                        flag=false;
                        e.printStackTrace();
                        session.getTransaction().rollback();                    //事务回滚
                }finally{
                        HibernateUtil.closeSession(session);                    //关闭 Session
                }
                return flag;
        }
```

秘笈心法

心法领悟 446：对象关联的使用。

对于建立关联的两个类，当需要通过关联操作映射对象时，要将两个类的对象建立关联。在本实例中，Depart 对象 depart 和 Employee 对象 em 之间在进行级联保存时，要通过 em.setDepart(depart)来实现两个对象的关联。

实例 447	建立商品表与商品类型表的双向关联 光盘位置：光盘\MR\17\447	中级 实用指数：★★✰

实例说明

本实例实现的是商品表与商品类型表的双向关联。运行实例，即可将商品表与商品类型表中相对应的数据

显示出来，如图 17.2 所示。

关键技术

要实现商品表与商品类型表的双向关联，就要建立从商品表到商品类型表的多对一关联，同时建立从商品类型表到商品表的一对多关联，这样就要配置<many-to-one>和<set>元素。下面分别介绍这两个元素的属性及其子元素。

（1）<many-to-one>元素

- name 属性：待映射的属性名称。
- column 属性：映射类对应表的外键。
- class 属性：name 属性值的类型。
- lazy 属性：设定对映射类是否采用延迟检索策略。

（2）<set>元素

- name 属性：待映射的属性名称。
- lazy 属性：设定对映射类是否采用延迟检索策略。
- <one-to-many>子元素的 class 属性：待映射类的名称，也表明在<set>元素的 name 属性值中存放的是一组 class 属性值类型的对象。

图 17.2　商品表与商品类型表的双向关联

（1）分别创建商品信息类和商品类型信息类，这两个类主要是定义商品和商品类型的一些属性。

（2）配置商品映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.Merchandise" table="tb_merchandise">
    <id name="id" column="id" type="int">                                        //定义表主键的映射
        <generator class="increment"/>
    </id>
    <property name="name" column="name" type="string" not-null="true"/>          //定义属性
    <property name="producingArea" column="producingArea" type="string" not-null="true"/>
    <property name="price" column="price" type="float" not-null="true"/>
    <many-to-one name="sort" column="sort_id" class="mrmwq.hibernate.Sort" lazy="false"/>   //定义<many-to-one>元素
</class>
```

（3）配置商品类型映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.Sort" table="tb_sort">                              //定义表主键的映射
    <id name="id" column="id" type="int">
        <generator class="increment"/>
    </id>
    <property name="name" column="name" type="string" not-null="true"/>
    <set name="merchandises" lazy="false">                                       //定义<set>元素
        <key column="sort_id"/>                                                  //定义子元素 key
        <one-to-many class="mrmwq.hibernate.Merchandise"/>                       //定义子元素<one-to-many>
    </set>
</class>
```

（4）创建操作数据库的 JSP 页面，同时把查询后的结果显示在页面上。具体代码如下：

```
<tr align="center">
    <td><table width="90%" border="1" cellspacing="0" cellpadding="4">
        <tr align="center" bgcolor="#FFCCFF">
            <td>序号</td>
            <td>商品名称</td>
            <td>商品种类</td>
            <td>产地</td>
            <td>价格</td>
        </tr>
        <%
        List l=h.query("from Sort ");                                            //查询商品类型表中所有数据
        for(int i=0;i<l.size();i++){
            Sort s=(Sort)l.get(i);                                               //得到每一个商品类型对象
```

```
            Set set=s.getMerchandises();                                                 //得到该对象对应的商品对象
            Iterator it=set.iterator();
            while(it.hasNext()){                                                         //遍历商品对象中的每一个属性并输出
               Merchandise m=(Merchandise)it.next();
         %>
         <tr align="center">
            <td><%=i+1 %></td>
            <td><%=m.getName() %></td>
            <td><%=m.getSort().getName() %></td>
            <td><%=m.getProducingArea() %></td>
            <td><%=m.getPrice() %></td>
         </tr>
         <%
            }
         }
         %>
      </table></td>
   </tr>
```

秘笈心法

心法领悟 447：<set>元素的 inverse 属性。

本实例中并没有配置 inverse 属性，在这种情况下 Hibernate 默认 inverse 属性值为 false；假如适当地设置 inverse 属性值为 true，则可以减少 Hibernate 执行 SQL 语句的条数，提高软件的运行效率。

实例 448 实现商品表的自关联

光盘位置：光盘\MR\17\448

中级

实用指数：★★☆

实例说明

本实例实现的是商品表的自关联。运行实例，即可看到“所属类别”列中显示的是属于哪一类的商品，如图 17.3 所示。

自关联查询

序号	商品类别	所属类别
0	食品	无
1	水果	无
2	服装	无
3	巧克力	食品
4	瓜子	食品
5	苹果	水果
6	香蕉	水果
7	裤子	服装
8	袜子	服装
9	篮球鞋	服装

图 17.3　商品表的自关联

关键技术

要实现商品表的自关联，首先要知道以下两点。

- 一种类别可以有父类别，也可以无父类别。
- 一种类别可以包含 0 个、1 个或多个子类别。

对于上述情况，通过 Hibernate 仍然可以建立关联关系，称为一对多自关联。这只是一对多关联关系的特殊情况，所以需要同时在一个映射文件中配置<set>和<many-to-one>元素。

（1）创建商品信息类，在该类中创建一个类型为 java.util.Set 的集合属性 sonSort，用来保存该类别所包含的子类别对象，同时建立一个该类的对象用来保存该类别所属的父类别对象。

（2）创建商品信息类的映射文件，具体代码如下：

```
<%
<class name="mrmwq.hibernate.Sort" table="tb_sortconn">                                              //定义表主键的映射
   <id name="id" column="id" type="int">
      <generator class="increment"/>
   </id>
   <property name="name" column="name" type="string" not-null="true"/>
   <many-to-one name="fatherSort" column="father_id" class="mrmwq.hibernate.Sort" lazy="false"/>     //定义<many-to-one>元素
   <set name="sonSorts" lazy="false">                                                                 //定义<set>元素
      <key column="father_id"/>
      <one-to-many class="mrmwq.hibernate.Sort"/>
```

```
    </set>
  </class>
```

（3）创建操作数据库的 JSP 页面，同时把查询后的结果显示在页面上。具体代码如下：

```
<tr align="center">
        <td><table width="35%"   border="1" cellspacing="0" cellpadding="4">
          <tr align="center" bgcolor="#FFCCFF">
            <td>序号</td>
            <td>商品类别</td>
            <td>所属类别</td>
          </tr>
          <%
          List l=h.query("from Sort");                                       //查询商品表中全部字段
          for(int i=0;i<l.size();i++){
            Sort s=(Sort)l.get(i);                                           //得到商品表中每一个对象
          %>
          <tr align="center">                                                //循环输出商品信息表中数据
            <td><%=i %></td>
            <td><%=s.getName() %></td>
            <%
            String fatherSort="无";
            if(s.getFatherSort()!=null){
              fatherSort=s.getFatherSort().getName();                        //得到父类对象的名称
            }
            %>
            <td><%=fatherSort %></td>
          </tr>
          <%
          }
          %>
        </table></td>
</tr>
```

秘笈心法

心法领悟 448：自关联的另一种实现。

本实例中只是实现了从子类获得父类中相应的信息，即一对多双向自关联中的多对一关联，读者可以自己编写一个从父类中获得子类信息的实例。



实例 449　在持久化类方法中加入程序代码

光盘位置：光盘\MR\17\449

中级

实用指数：★★☆

实例说明

本实例实现的是在持久化类方法中加入员工职务级别程序代码。运行实例，在新建员工档案页面中输入相应的信息，同时选择职务级别，单击“提交”按钮，即可把职务级别映射为数据库中的一个字段，然后显示在页面上，如图 17.4 所示。



图 17.4　在持久化类方法中加入程序代码



要实现在持久化类方法中加入程序代码，就要在持久化类中定义两个派生属性 job、level 以及相应的 get()、set()方法，分别用来接收前台传来的职务和级别。之所以称为派生属性，是因为在映射文件中并没有将其直接映射到数据表。同时，也要在持久化类中定义 duty 属性，并提供对应的 get()、set()方法。在检索数据时，Hibernate 会调用相应的 set()方法，set()方法通过处理传入的值，将派生

属性 job 和 level 初始化。在持久化数据时，Hibernate 会调用相应的 get()方法，完成持久化操作。

设计过程

（1）创建员工信息类，在该类中主要是定义员工的普通属性和派生属性 job、level。

（2）创建员工信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.Staff" table="tb_staff">
    <id name="id" column="id" type="int" access="field">                    //定义表主键的映射
        <generator class="increment"/>
    </id>
    <property name="name" column="name" type="string" not-null="true"/>      //定义属性
    <property name="sex" column="sex" type="string" not-null="true"/>
    <property name="birthday" column="birthday" type="string" not-null="true"/>
    <property name="duty" column="duty" type="string" not-null="true"/>
</class>
```

（3）创建操作数据库的 JSP 页面，同时把查询后的结果显示在页面上。具体代码如下：

```
<tr align="center">
        <td><table width="90%" border="1" cellspacing="0" cellpadding="4">
          <tr align="center" bgcolor="#FFCCFF">
            <td>序号</td>
            <td>姓名</td>
            <td>性别</td>
            <td>出生日期</td>
            <td>职务级别</td>
          </tr>
          <%
          List l=h.query("from Staff where duty is not null");              //查询数据库中信息
          for(int i=0;i<l.size();i++){
            Staff s=(Staff)l.get(i);                                        //得到每一个员工对象
          %>
          <tr align="center">                                               //输出每一个员工信息
            <td><%=i+1 %></td>
            <td><%=s.getName() %></td>
            <td><%=s.getSex() %></td>
            <td><%=s.getBirthday() %></td>
            <td><%=s.getDuty()%></td>
          </tr>
          <%
          }
          %>
        </table></td>
</tr>
```

秘笈心法

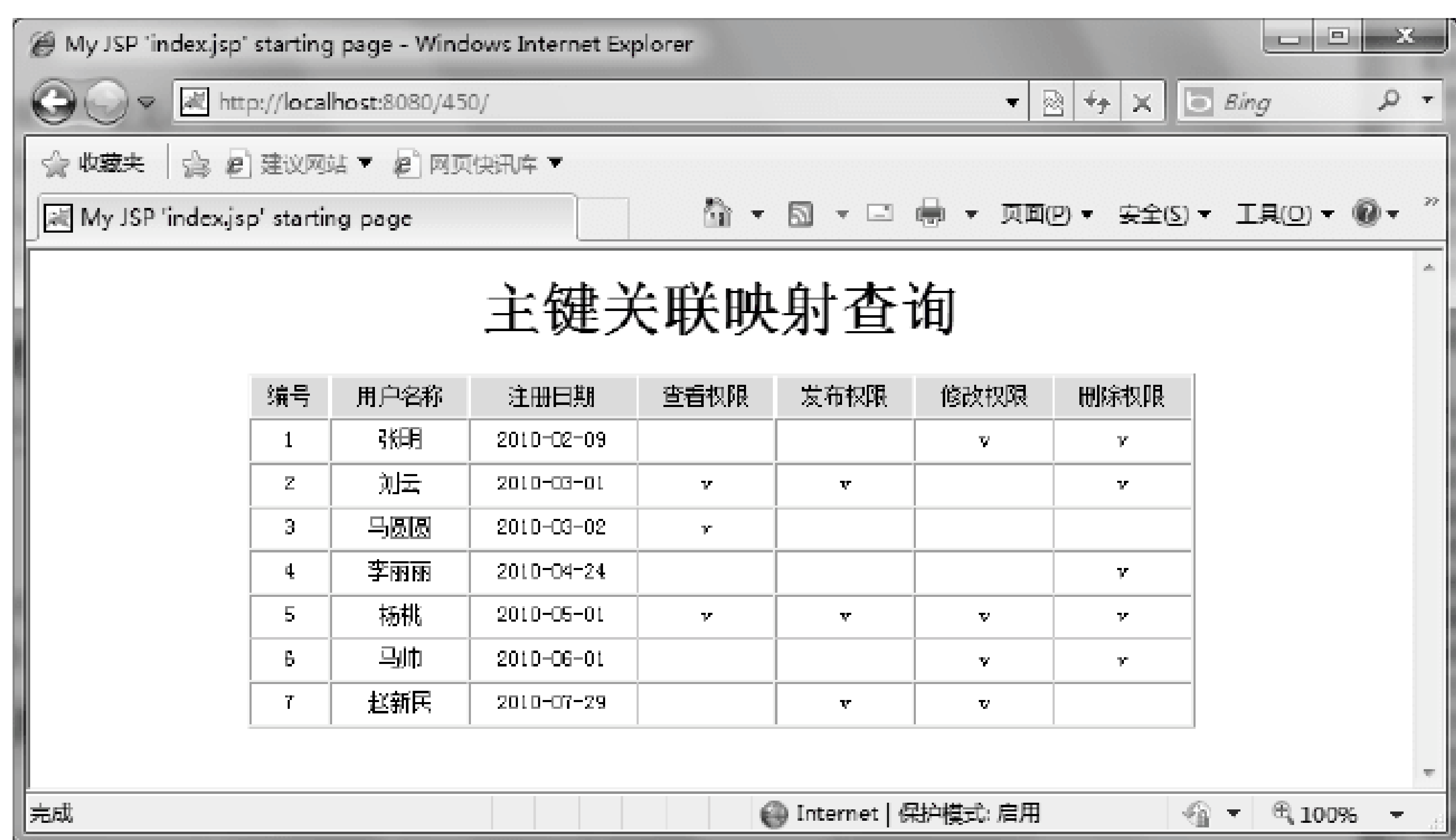
心法领悟 449：duty 属性的定义。

在本实例中并不需要定义 duty 属性，原因有二。一是因为映射文件中并没有设置<property>元素的 access 属性，在默认情况下，Hibernate 会直接访问 duty 属性对应的 get()、set()方法；二是因为此处不需要用 duty 属性来保持员工的基本信息。

实例 450	主键关联映射 光盘位置：光盘\MR\17\450	中级 实用指数：★★☆

实例说明

本实例实现的是主键关联映射查看用户权限。运行实例，即可查看用户对各个操作的权限，其中显示 v 的表示有权限执行该操作，显示为空的表示没有权限执行该操作，如图 17.5 所示。



编号	用户名称	注册日期	查看权限	发布权限	修改权限	删除权限
1	张明	2010-02-09			v	v
2	刘云	2010-03-01	v	v		v
3	马圆圆	2010-03-02	v			
4	李丽丽	2010-04-24				v
5	杨桃	2010-05-01	v	v	v	v
6	马帅	2010-06-01			v	v
7	赵新民	2010-07-29		v	v	

图 17.5　主键的关联映射

关键技术

要实现主键关联映射查看用户权限，在两个类相对应的映射文件中都要使用<one-to-one>元素。该元素的主要属性介绍如下。

- name：欲映射的实例名称。
- class：欲映射的实例类型。
- cascade：设定级联操作的权限。本实例中采用级联操作的权限为 all。
- lazy：设定是否采用延迟加载。本实例中并没有采用延迟加载。
- constrained：该属性设置为 true，表示该映射文件所映射表的主键同时作为外键，参照关联类对应表的主键。

（1）创建用户信息类和权限信息类，这两个类主要用于定义用户信息和权限信息的一些属性。

（2）创建用户信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.UserInfo" table="tb_userInfo">
    <id name="id" column="id" type="int" access="field">                                   //定义表主键的映射
      <generator class="increment"/>
    </id>
    <property name="username" column="username" type="string" not-null="true"/>            //定义类中属性
    <property name="password" column="password" type="string" not-null="true"/>
    <property name="loginTime" column="loginTime" type="date" not-null="true"/>
    <one-to-one name="purviewPK" class="mrmwq.hibernate.PurviewPK" cascade="all" lazy="false"/>   //定义<one-to-one>元素
</class>
```

（3）创建权限信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.PurviewPK" table="tb_purviewPK">
    <id name="id" column="id" type="int" access="field">                  //定义表主键的映射
      <generator class="foreign">                                          //定义该主键同时为外键
        <param name="property">userInfo</param>                            //设定外键的参考信息
      </generator>
    </id>
    <property name="see" column="see" not-null="true"/>                    //定义类中属性
    <property name="ist" column="ist" not-null="true"/>
    <property name="udt" column="udt" not-null="true"/>
    <property name="dlt" column="dlt" not-null="true"/>
    <one-to-one name="userInfo" class="mrmwq.hibernate.UserInfo" constrained="true" lazy="false"/>   //定义<one-to-one>元素
</class>
```

（4）创建操作数据库的 JSP 页面，同时把查询后的结果显示在页面上。具体代码如下：

```
<tr align="center">
        <td><table width="90%" border="1" cellspacing="0" cellpadding="4">
          <tr align="center" bordercolor="#FFFFFF" bgcolor="#FFCCFF">
            <td>编号</td>
```

```
            <td>用户名称</td>
            <td>注册日期</td>
            <td>查看权限</td>
            <td>发布权限</td>
            <td>修改权限</td>
            <td>删除权限</td>
          </tr>
          <%
          List l=h.query("from UserInfo");                                          //查询数据库中所有信息
          for(int i=0;i<l.size();i++){
            UserInfo u=(UserInfo)l.get(i);                                          //得到每一个用户对象
          %>
          <tr align="center">
            <td><%=u.getId() %></td>                                                //循环输出每一个用户信息和权限信息
            <td><%=u.getUsername() %></td>
            <td><%=u.getLoginTime() %></td>
            <td><%if(u.getPurviewPK().isSee()){out.print("v");}else{out.print(" ");} %></td>
            <td><%if(u.getPurviewPK().isIst()){out.print("v");}else{out.print(" ");} %></td>
            <td><%if(u.getPurviewPK().isUdt()){out.print("v");}else{out.print(" ");} %></td>
            <td><%if(u.getPurviewPK().isDlt()){out.print("v");}else{out.print(" ");} %></td>
          </tr>
          <%
          }
          %>
        </table></td>
</tr>
```

秘笈心法

心法领悟 450：配置映射文件中 OID 的生成方式。

因为映射文件 PurviewPK.hhm.xml 所映射的主键同时也作为外键，所以需要在配置文件中配置 OID 的生成方式。其中需要配置两个元素，第一个是<generator>元素，将该元素的 class 属性设置为 foreign，说明该主键同时为外键，所以主键的生成方式将参考外键；另一个是<param>元素，该元素用来设定外键的参考信息。

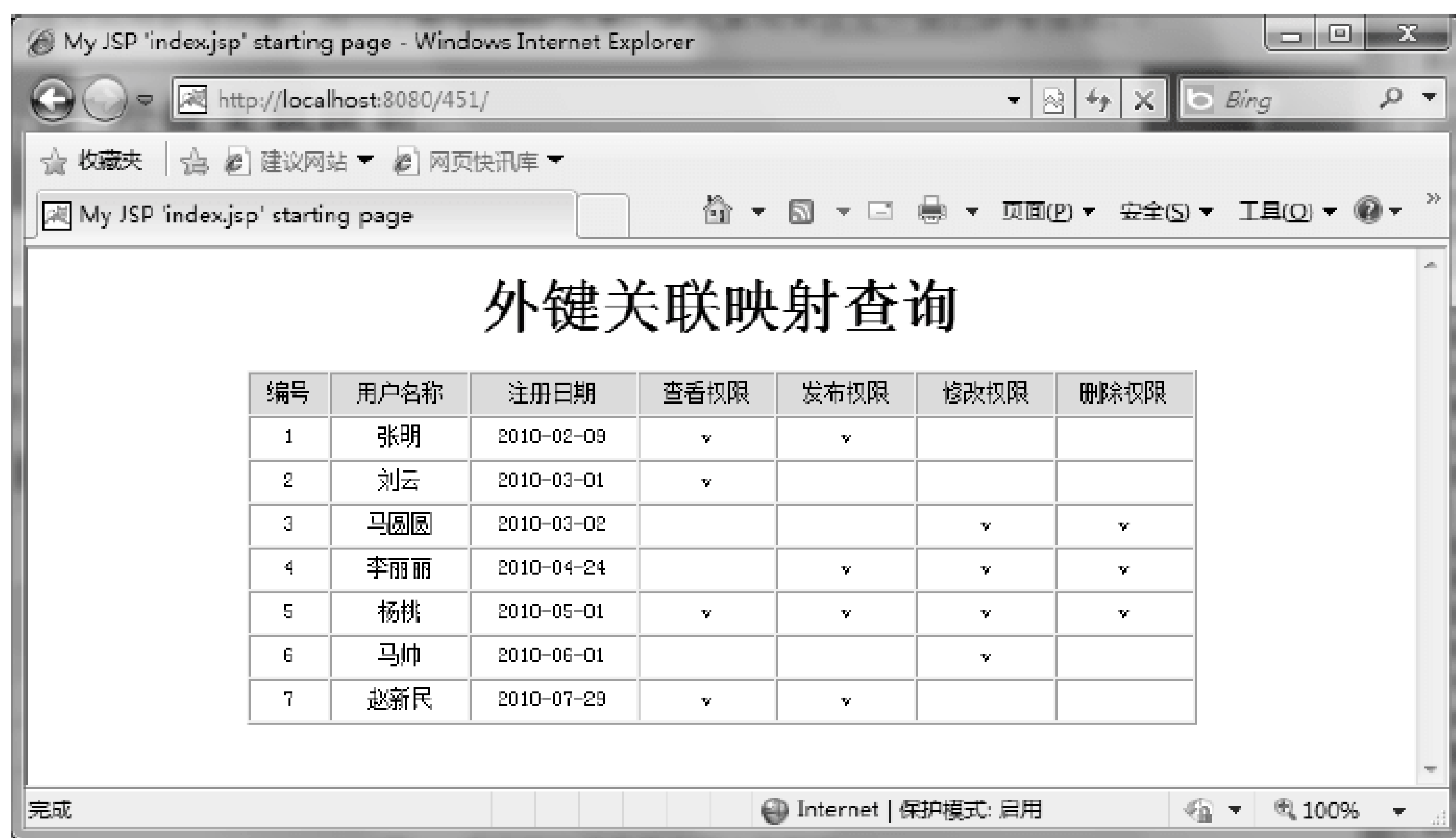
实例 451 外键关联映射

光盘位置：光盘\MR\17\451

中级

实用指数：★★☆

本实例实现的是外键关联映射查看用户权限。运行实例，即可查看用户对各个操作的权限，其中显示 v 的表示有权限执行该操作，显示为空的表示没有权限执行该操作，如图 17.6 所示。



编号	用户名称	注册日期	查看权限	发布权限	修改权限	删除权限
1	张明	2010-02-09	v	v		
2	刘云	2010-03-01	v			
3	马圆圆	2010-03-02			v	v
4	李丽丽	2010-04-24		v	v	v
5	杨桃	2010-05-01	v	v	v	v
6	马帅	2010-06-01			v	
7	赵新民	2010-07-29	v	v		

图 17.6　外键关联映射

关键技术

要实现外键关联映射查看用户权限，需在两个类相对应的映射文件中一方使用<one-to-one>元素，另一方使用<many-to-one>元素。下面分别对这两个元素的属性进行介绍。

（1）<one-to-one>元素。

- name：欲映射的实例名称。
- class：欲映射的实例类型。
- cascade：设定级联操作的权限，本实例中采用级联操作的权限为 all。
- lazy：设定是否采用延迟加载，本实例中并没有采用延迟加载。

（2）<many-to-one>元素。

- name：欲映射的实例名称。
- class：欲映射的实例类型。
- column：关联表的外键
- lazy：设定是否采用延迟加载，本实例中并没有采用延迟加载。
- unique：unique 等于 true，表示该映射文件所映射类的实例只与一个关联类的实例对应。

（1）创建用户信息类和权限信息类，这两个类主要是定义用户信息和权限信息的一些属性。

（2）创建用户信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.UserInfo" table="tb_userInfo">
    <id name="id" column="id" type="int" access="field">                                    //定义表主键的映射
        <generator class="increment"/>
    </id>
    <property name="username" column="username" type="string" not-null="true"/>            //定义类中属性
    <property name="password" column="password" type="string" not-null="true"/>
    <property name="loginTime" column="loginTime" type="date" not-null="true"/>
    <one-to-one name="purviewFK" class="mrmwq.hibernate.PurviewFK" cascade="all" lazy="false"/>   //定义<one-to-one>元素
</class>
```

（3）创建权限信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.PurviewFK" table="tb_purviewFK">
    <id name="id" column="id" type="int" access="field">                                    //定义表主键的映射
        <generator class="increment"/>
    </id>
    <property name="see" column="see" type="boolean" not-null="true"/>                     //定义类的属性
    <property name="ist" column="ist" type="boolean" not-null="true"/>
    <property name="udt" column="udt" type="boolean" not-null="true"/>
    <property name="dlt" column="dlt" type="boolean" not-null="true"/>
    <many-to-one name="userInfo" class="mrmwq.hibernate.UserInfo" column="user_id" unique="true" lazy="false"/>   //定义<many-to-one>元素
</class>
```

（4）创建操作数据库的 JSP 页面，同时把查询后的结果显示在页面上，具体代码如下：

```
<tr align="center">
        <td><table width="90%"   border="1" cellspacing="0" cellpadding="4">
          <tr align="center" bgcolor="#FFCCFF">
            <td>编号</td>
            <td>用户名称</td>
            <td>注册日期</td>
            <td>查看权限</td>
            <td>发布权限</td>
            <td>修改权限</td>
            <td>删除权限</td>
          </tr>
          <%
          List l=h.query("from UserInfo");                          //查询数据库中所有信息
          for(int i=0;i<l.size();i++){
            UserInfo u=(UserInfo)l.get(i);                          //得到每一个用户对象
          %>
```

```
            <tr align="center">
              <td><%=u.getId() %></td>                                                    //循环输出每一个用户信息和权限信息
              <td><%=u.getUsername() %></td>
              <td><%=u.getLoginTime() %></td>
              <td><%if(u.getPurviewFK().isSee()){out.print("v");}else{out.print(" ");} %></td>
              <td><%if(u.getPurviewFK().isIst()){out.print("v");}else{out.print(" ");} %></td>
              <td><%if(u.getPurviewFK().isUdt()){out.print("v");}else{out.print(" ");} %></td>
              <td><%if(u.getPurviewFK().isDlt()){out.print("v");}else{out.print(" ");} %></td>
            </tr>
            <%
            }
            %>
          </table></td>
</tr>
```

秘笈心法

心法领悟 451：外键关联的解释。

所谓外键关联，是指从表的主键并不作为外键参考主表的主键，而是将其他字段作为外键参考主表的主键。以外键的方式可以建立各种关联关系。

实例 452 多对多单向关联映射学生表与科目表

光盘位置：光盘\MR\17\452

中级

实用指数：★★☆

实例说明

本实例实现的是多对多单向关联映射学生表与科目表。运行实例，可以看到一个学生可以选修多个课程，同时一个课程可以被多个学生所选修，如图 17.7 所示。

多对多的单向关联

姓名	性别	专业	选修科目
张云	男	计算机	Java基础教程
马良	男	英语	Java基础教程
杨桃	女	计算机	计算机组成原理
温明远	男	政治	JavaScript网页编程
刘晁	男	政治	Java基础教程
赵婷婷	女	英语	Html技术内幕　Linux开发大全
卿明祥	男	计算机	Java基础教程　计算机组成原理

图 17.7　多对多单向关联映射学生表与科目表

关键技术

要实现多对多的单向关联，可以通过 Hibernate 中的<set>元素来实现。该元素的属性及其子元素分别介绍如下。

- name 属性：欲映射的 Set 类型的属性名称。
- table 属性：关联表的名称。
- <key>元素的 column 属性：关联表中以该映射文件所映射表的主键为外键的字段名称。
- <many-to-many>元素的 class 属性：关联类的名称，同时也说明在 Set 类型的属性中存放的为该类的实例。
- <many-to-many>元素的 column 属性：关联表中以欲关联类对应表的主键为外键的字段名称。

（1）分别创建学生信息类和科目信息类，这两个类主要是定义学生和科目的一些属性。

（2）创建学生信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.Student" table="tb_student">
  <id name="id" column="id" type="int" access="field">                                        //定义表主键的映射
    <generator class="increment"/>
  </id>
  <property name="name" column="name" type="string" not-null="true"/>                          //定义类的属性
  <property name="sex" column="sex" type="string" not-null="true"/>
  <property name="specialty" column="specialty" type="string" not-null="true"/>
  <set name="subjects" table="tb_student_subject" lazy="false" cascade="save-update">          //定义<set>元素
    <key column="student_id"/>
```

```
        <many-to-many class="mrmwq.hibernate.Subject" column="subject_id"/>
    </set>
</class>
```

（3）创建科目信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.Subject" table="tb_subject">
    <id name="id" column="id" type="int" access="field">                     //定义表主键映射
        <generator class="increment"/>
    </id>
    <property name="name" column="name" type="string" not-null="true"/>      //定义类属性
</class>
```

（4）创建操作数据库的 JSP 页面，同时把查询后的结果显示在页面上。具体代码如下：

```
<tr align="center">
            <td><table width="95%"   border="1" cellspacing="0" cellpadding="2">
                <tr align="center" bgcolor="#FFCCFF">
                    <td>姓名</td>
                    <td>性别</td>
                    <td>专业</td>
                    <td>选修科目</td>
                </tr>
                <%
                    List l=h.query("from Student");                          //查询数据库中所有信息
                    for(int i=0;i<l.size();i++){
                    Student st=(Student)l.get(i);                            //得到每一个 student 对象
                    Set s=st.getSubjects();                                  //得到每一个 student 对象对应的科目
                    Iterator it=s.iterator();
                    String subjects="";
                    while(it.hasNext()){
                    Subject su=(Subject)it.next();                           //得到 set 集合中的每一个科目对象
                    subjects=subjects+su.getName()+"  ";           //将得到的科目名称进行拼接
        }
                %>
                <tr align="center">
                    <td><%=st.getName() %></td>                              //循环输出信息
                    <td><%=st.getSex() %></td>
                    <td><%=st.getSpecialty() %></td>
                    <td align="left"><%=subjects %></td>
                </tr>
                <%
          }
          %>
            </table></td>
</tr>
```

秘笈心法

心法领悟 452：多对多关联时 cascade 属性的设置。

在建立多对多关联时，建议将 cascade 属性设置为 save-update，因为通常都需要级联保存和修改关联对象；但是不可以设置为包含级联删除关联对象权限的值，因为关联对象还可能与其他对象存在关联关系，具体设置还要根据实际情况而定。

实例 453	多对多双向关联映射学生表与科目表 光盘位置：光盘\MR\17\453	中级 实用指数：★★☆

实例说明

本实例实现的是多对多双向关联映射学生表与科目表。运行实例，即可看到学生姓名以及选修课程都是从数据库中查找出的字段，在下拉列表框中选择学生姓名，同时在“选修课程”选项组中选择选修课程，单击“确定”按钮，即可成功选修课程，如图 17.8 所示。

添加选修信息

学生姓名：张云

选修课程：

☐ Java基础教程
☐ 计算机组成原理
☑ Linux开发大全
☑ Html技术内幕
☐ JavaScript网页编程
☐ 高等数学

确定 重置

图 17.8　多对多双向关联映射

关键技术

在建立多对多的双向关联时，关联的两端都要通过<set>元素映射关联关系。在这种情况下，必须把其中一端的<set>元素的 inverse 设置为 true。对于 inverse 为 true 的一端，可以通过<bag>元素实现映射；对于 inverse 为 false 的一端，可以通过<idbag>和<list>元素实现映射。

（1）分别创建学生信息类和科目信息类，这两个类主要是定义学生和科目的一些属性。

（2）创建学生信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.Student" table="tb_student">
  <id name="id" column="id" type="int" access="field">                                    //定义表主键的映射
    <generator class="increment"/>
  </id>
  <property name="name" column="name" type="string" not-null="true"/>                    //定义类的属性
  <property name="sex" column="sex" type="string" not-null="true"/>
  <property name="specialty" column="specialty" type="string" not-null="true"/>
  <set name="subjects" table="tb_student_subject" lazy="false" cascade="save-update">    //定义<set>元素
    <key column="student_id"/>
    <many-to-many class="mrmwq.hibernate.Subject" column="subject_id"/>
  </set>
</class>
```

（3）创建科目信息类的映射文件，具体代码如下：

```
 <class name="mrmwq.hibernate.Subject" table="tb_subject">
  <id name="id" column="id" type="int" access="field">                                    //定义表主键的映射
   <generator class="increment"/>
  </id>
  <property name="name" column="name" type="string" not-null="true"/>                    //定义类的属性
  <set name="students" table="tb_student_subject" lazy="false" cascade="save-update" inverse="true">   //定义<set>元素
    <key column="subject_id"/>
    <many-to-many class="mrmwq.hibernate.Student" column="student_id"/>
  </set>
</class>
```

（4）创建操作数据库的 JSP 页面，同时把前台提交的数据更新到数据库中，然后把查询后的选修课程显示在页面上。具体代码如下：

```
<%
  String stId=request.getParameter("studentId");                          //获取前台提交的学生 id
  Student st=(Student)h.queryOne("from Student where id="+stId);         //查询指定 id 的学生对象
  st.getSubjects().clear();
  h.update(st);                                                           //更新该对象
  String[] suIdS=request.getParameterValues("subjectId");                 //获取前台提交的课程 id
  String suId="";
  for(int i=0;i<suIdS.length;i++){
    suId=suId+","+suIdS[i];                                               //将课程 id 进行拼接
  }
  List suS=h.query("from Subject where id in("+suId.substring(1)+")");   //查询指定学生的课程对象
  for(int i=0;i<suS.size();i++){
    Subject su=(Subject)suS.get(i);
    su.getStudents().add(st);                                             //添加指定学生对象
    st.getSubjects().add(su);                                             //添加指定课程对象
```

```
    }
    h.update(st);                                                                    //更新学生对象
    %>
        <span class="STYLE8">选修信息添加成功</span></p>
    <p align="center">
      <%
          st=(Student)h.queryOne("from Student where id="+stId);                      //查询指定 id 的学生对象
          String subjects="";
          Set suSet=st.getSubjects();
          Iterator suIt=suSet.iterator();
          while(suIt.hasNext()){
            Subject su=(Subject)suIt.next();
            subjects=subjects+"、"+su.getName();
          }
          out.print(st.getName()+"   选修的课程有"+subjects.substring(1)+"!");           //循环输出选修课程
%>
```

秘笈心法

心法领悟 453：<key>元素和<many-to-many>元素的 column 属性。

在刚开始编写映射文件时，很容易把这两个元素的 column 属性弄混。读者需要牢记以下原则：<key>元素的 column 属性映射的是关联表中以所在映射文件对应表的主键为外键的字段名称，<many-to-many>元素的 column 属性映射的是关联表中以欲关联表的主键为外键的字段名称。

17.2　Hibernate 检索策略

实例 454	一对多的立即检索策略 光盘位置：光盘\MR\17\454	中级 实用指数：★★☆

实例说明

本实例实现的是立即检索公司名称和部门信息。运行实例，即可看到控制台上打印出两条 select 语句和公司名称以及部门名称，如图 17.9 所示。

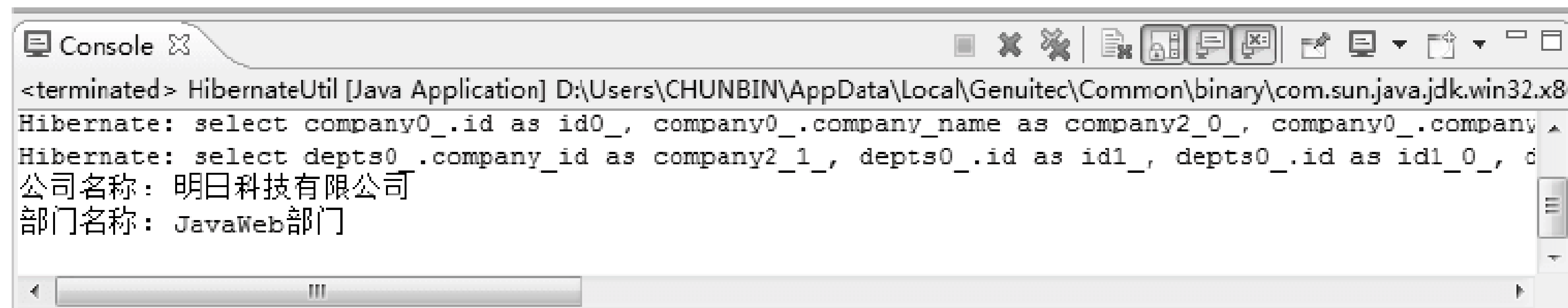


图 17.9　一对多的立即检索策略

关键技术

对于一对多和多对一的立即检索策略要注意以下两点：

- 通常情况下，在一对多关联中应尽量避免使用立即检索策略，特殊情况除外。
- 在多对一关联中，应根据实际情况决定采用何种检索策略。

Hibernate 默认一对多关联采用延迟检索策略，因为通常情况下，在程序中都不需要立即访问关联对象，甚至不需要访问关联对象，如果在这种情况下采用立即检索策略，无疑会执行多余的 select 语句，延迟加载时间，并且还会加载不需要的对象，占用缓存空间。

（1）创建部门信息类和公司信息类，这两个类主要用于定义部门和公司的一些属性。

（2）创建部门信息类的映射文件，具体代码如下：

```
 <class name="mrmwq.hibernate.Dept" table="tb_dept">
  <id name="id" column="id" type="int">                                          //定义表主键的映射
    <generator class="increment"/>
  </id>
  <property name="company_id" column="company_id" type="int" not-null="true"/>   //定义类的属性
  <property name="name" column="name" type="string" not-null="true"/>
</class>
```

（3）创建公司信息类的映射文件，具体代码如下：

```
  <class name="mrmwq.hibernate.Company" table="tb_company">
  <id name="id" column="id" type="int">                                                     //定义表主键的映射
    <generator class="increment"/>
  </id>
  <property name="company_name" column="company_name" type="string" not-null="true"/>      //定义类的属性
  <property name="company_type" column="company_type" type="string" not-null="true"/>
  <property name="owner" column="owner" type="string" not-null="true"/>
  <property name="enrol_fund" column="enrol_fund" type="int" not-null="true"/>
  <property name="enrol_date" column="enrol_date" type="date" not-null="true"/>
  <set name="depts" table="tb_dept" cascade="all" lazy="false">                           //设置<set>元素并采用立即检索策略
    <key column="company_id"/>
    <one-to-many class="mrmwq.hibernate.Dept"/>
  </set>
  </class>
```

（4）创建 HibernateUtil 类，该类主要用于对数据进行查询并输出到控制台。

```
public static void main(String[] args) {
      HibernateUtil h = new HibernateUtil();
      Company com = (Company) h.openSession().createQuery(                 //查询指定 id 的 Company 对象
              "from Company where id=1").uniqueResult();
     h.closeSession();                                                     //关闭 Session
      System.out.println("公司名称：  " +
                          com.getCompany_name());                          //输出公司名称
      Iterator it = com.getDepts().iterator();
      while (it.hasNext()) {
          Dept d = (Dept) it.next();
          System.out.println("部门名称：  " + d.getName());                 //输出部门名称
      }
 }
```

秘笈心法

心法领悟 454：运行结果的进一步解释。

在运行结果中输出了两条 select 语句，这两条语句是在执行查询时生成的，并且初始化了全部对象，当 Company 对象调用属性时，并没有执行 select 语句。

实例 455	多对一的立即检索策略 光盘位置：光盘\MR\17\455	中级 实用指数：★★☆

本实例实现的也是立即检索公司名称和部门信息，只不过采用的是多对一立即检索策略。运行实例，即可看到控制台上打印出两条 select 语句和公司名称以及部门名称，如图 17.10 所示。

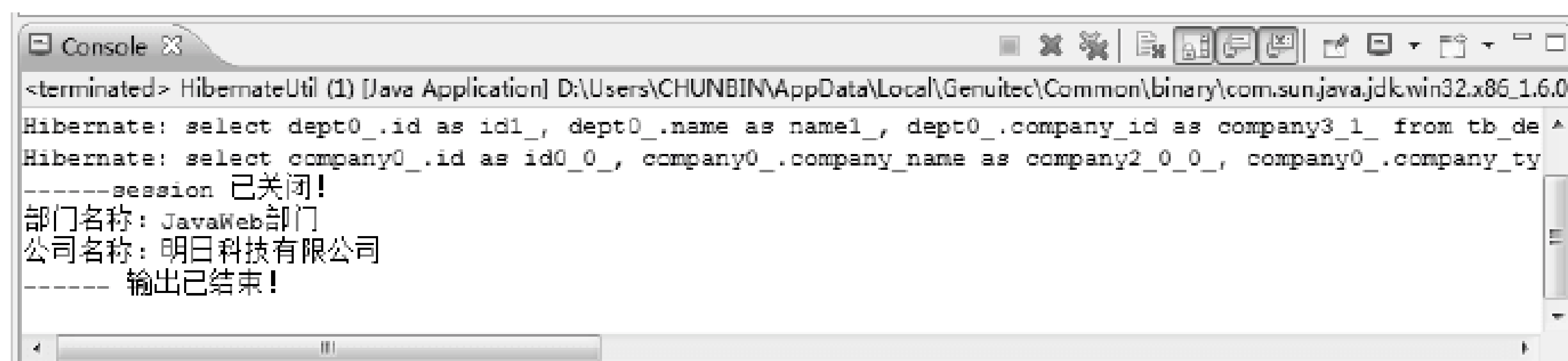


图 17.10　多对一的立即检索策略

关键技术

Hibernate 多对一关联默认也采用延迟检索策略。一般情况下，采用立即检索策略并不会对性能产生太大的影响，因为在“多”的那一端的一个对象只有一个关联对象与之相对应。不过，还是要根据实际情况决定采用何种检索策略。例如，每次或者多数情况下在加载当前对象时，都需要立即访问与之关联的对象，则可以采用立即检索策略。

要实现立即检索策略，需要把<many-to-one>元素的 lazy 属性设置为 false。

设计过程

（1）创建部门信息类和公司信息类，这两个类主要用于定义部门和公司的一些属性。

（2）创建部门信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.Dept" table="tb_dept">
  <id name="id" column="id" type="int">                                                    //定义表主键的映射
    <generator class="increment"/>
  </id>
  <property name="name" column="name" type="string" not-null="true"/>
  <many-to-one name="company" column="company_id" class="mrmwq.hibernate.Company" lazy="false"/> //定义<many-to-one>元素并立即检索
  </class>
```

（3）创建公司信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.Company" table="tb_company">
  <id name="id" column="id" type="int">                                                    //定义表主键的映射
    <generator class="increment"/>
  </id>
  <property name="company_name" column="company_name" type="string" not-null="true"/>      //定义类属性
  <property name="company_type" column="company_type" type="string" not-null="true"/>
  <property name="owner" column="owner" type="string" not-null="true"/>
  <property name="enrol_fund" column="enrol_fund" type="int" not-null="true"/>
  <property name="enrol_date" column="enrol_date" type="date" not-null="true"/>
  </class>
```

（4）创建 HibernateUtil 类，该类主要是对数据进行查询并输出到控制台。

```
public static void main(String[] args) {
      HibernateUtil h = new HibernateUtil();
      List l=h.openSession().createQuery("from Dept where name like '%明日%'").list();   //公司名称模糊查询
      h.closeSession();                                                                //关闭 Session
      System.out.println("------ Session 已关闭！ ");
      for(int i=0;i<l.size();i++){
            Dept d = (Dept) l.get(i);
            Company com = d.getCompany();                                              //输出公司名称
            System.out.println("公司名称："+com.getCompany_name());
      }
      System.out.println("------  输出已结束！ ");
 }
```

秘笈心法

心法领悟 455：lazy 属性的设置。

在本例中若没有设置 lazy 属性，Hibernate 将默认采用延迟检索策略，此时再运行该程序就会出错，控制台会打印出“could not initialize proxy - no Session”的错误。

实例 456	一对多的延迟检索策略 光盘位置：光盘\MR\17\456	中级 实用指数：★★✰

实例说明

本实例实现的是延迟检索公司名称和部门信息。运行实例，即可看到控制台上打印出两条 select 语句和公

司名称以及部门名称，如图 17.11 所示。

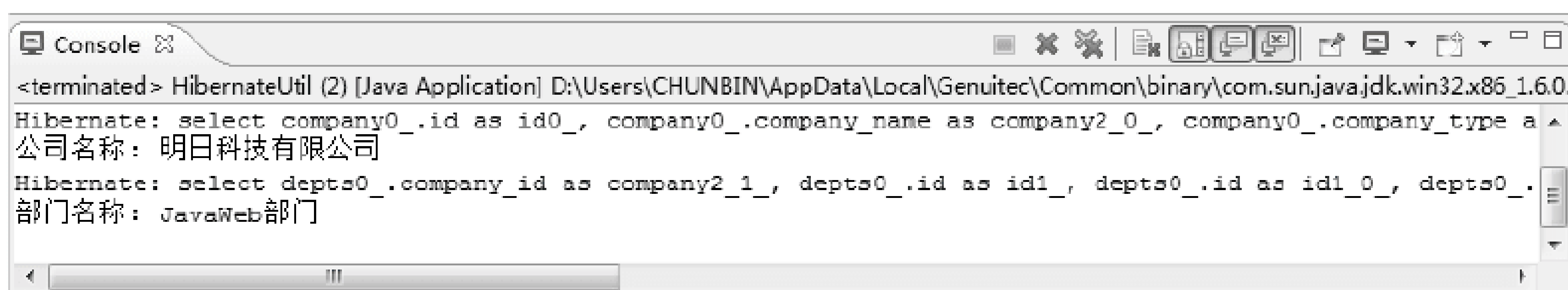


图 17.11　一对多的延迟检索策略

关键技术

对于一对多和多对一的延迟检索策略要注意以下两点：

- 通常情况下，在一对多关联中应尽量使用延迟检索策略，特殊情况除外。
- 在多对一关联中，应根据实际情况决定采用何种检索策略。

Hibernate 默认一对多关联采用延迟检索策略。当采用延迟检索策略时，在执行检索时并不初始化关联对象，只是创建关联对象的代理对象，这些代理对象只有 OID 被初始化，只有在程序中访问关联对象时，才对关联对象进行初始化。

（1）创建部门信息类和公司信息类，这两个类主要用于定义部门和公司的一些属性。

（2）创建部门信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.Dept" table="tb_dept">
  <id name="id" column="id" type="int">                                    //定义表主键映射
    <generator class="increment"/>
  </id>
  <property name="company_id" column="company_id" type="int" not-null="true"/>    //定义类的属性
  <property name="name" column="name" type="string" not-null="true"/>
</class>
```

（3）创建公司信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.Company" table="tb_company">
  <id name="id" column="id" type="int">                                    //定义表主键映射
    <generator class="increment"/>
  </id>
  <property name="company_name" column="company_name" type="string" not-null="true"/>    //定义类的属性
  <property name="company_type" column="company_type" type="string" not-null="true"/>
  <property name="owner" column="owner" type="string" not-null="true"/>
  <property name="enrol_fund" column="enrol_fund" type="int" not-null="true"/>
  <property name="enrol_date" column="enrol_date" type="date" not-null="true"/>
  <set name="depts" table="tb_dept" cascade="all" >                        //设置<set>元素并采用延迟检索策略
    <key column="company_id"/>
    <one-to-many class="mrmwq.hibernate.Dept"/>
  </set>
</class>
```

（4）创建 HibernateUtil 类，该类主要是对数据进行查询并输出到控制台。

```
public static void main(String[] args) {
        HibernateUtil h = new HibernateUtil();
        Company com = (Company) h.openSession().createQuery(                //查询指定 id 的 Company 对象
                "from Company where id=1").uniqueResult();
        h.closeSession();                                                   //关闭 Session
        System.out.println("公司名称：  " +
                        com.getCompany_name());                             //输出公司名称
        Iterator it = com.getDepts().iterator();
        while (it.hasNext()) {
            Dept d = (Dept) it.next();
            System.out.println("部门名称：  " + d.getName());                //输出部门名称
        }
  }
```

秘笈心法

心法领悟 456：运行结果的进一步解释。

在运行结果中同样输出了两条 select 语句，但是这两条 select 语句的输出顺序与前面实例有所不同，这是因为在执行检索时只执行了一条 select 语句初始化了公司对象，在输出部门信息时才执行初始化部门对象的 select 语句。

实例 457	迫切左外连接查询 光盘位置：光盘\MR\17\457	中级 实用指数：★★☆

实例说明

本实例实现的是迫切左外连接查询部门和公司信息。运行程序，将会看到无论是从公司到部门还是从部门到公司，其详细信息都被列出来并显示在页面上，如图 17.12 所示。

从公司到部门

公司名称：明日科技有限公司
公司类型：软件
公司法人：经理
注册资金：3000
注册时间：1998-09-06
部门名称：JavaWeb部门

从部门到公司

部门名称：JavaWeb部门
公司名称：明日科技有限公司
公司类型：软件
公司法人：经理
注册资金：3000
注册时间：1998-09-06

图 17.12　迫切左外连接查询

关键技术

迫切左外连接检索策略与立即检索策略实现的功能是相同的，区别在于前者只需要执行一条 select 语句，而后者需要执行若干条 select 语句，相对而言减少了访问数据库的次数，不过单次执行 select 语句的时间有所增加，因为 select 语句的复杂度增加了。从理论上讲，还是可以缩短总的访问时间的，不过这也不是绝对的。建议在以下情况下考虑使用该策略，前提是在最近一段时间内需要访问关联对象。

- 在一对一关联中。
- 在多对一关联中。
- 在一对多或者多对多关联中，附加条件是关联对象不要过多。

迫切左外连接检索策略通过配置映射文件的 fetch 属性来实现，在<many-to-one>、<one-to-one>和<set>元素中都含有该属性。

设计过程

（1）创建部门信息和公司信息类，这两个类主要用于定义部门和公司的一些属性。

（2）创建部门信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.Dept" table="tb_dept">
  <id name="id" column="id" type="int">                                              //定义表主键的映射
    <generator class="increment"/>
  </id>
  <property name="name" column="name" type="string" not-null="true"/>
  <many-to-one name="company" column="company_id" class="mrmwq.hibernate.Company" fetch="join"/> //配置 fetch 属性
</class>
```

（3）创建公司信息类的映射文件，具体代码如下：

```
  <class name="mrmwq.hibernate.Company" table="tb_company">
  <id name="id" column="id" type="int">
    <generator class="increment"/>                                                   //定义表主键的映射
  </id>
  <property name="company_name" column="company_name" type="string" not-null="true"/>         //定义类属性
  <property name="company_type" column="company_type" type="string" not-null="true"/>
  <property name="owner" column="owner" type="string" not-null="true"/>
  <property name="enrol_fund" column="enrol_fund" type="int" not-null="true"/>
  <property name="enrol_date" column="enrol_date" type="date" not-null="true"/>
  <set name="depts" table="tb_dept" cascade="all" fetch="select">                              //配置 fetch 属性
```

```
        <key column="company_id"/>
        <one-to-many class="mrmwq.hibernate.Dept"/>
    </set>
</class>
```

（4）创建操作数据库的 JSP 页面，同时把查询后的结果显示在页面上。具体代码如下：

```
<%
        h.openSession();
        Company com = h.getCompany(1);                                    //查询指定 id 的公司对象
        %>
        <tr><td>公司名称：<%=com.getCompany_name() %></td></tr>          //打印基本信息
        <tr><td>公司类型：<%=com.getCompany_type() %></td></tr>
        <tr><td>公司法人：<%=com.getOwner() %></td></tr>
        <tr><td>注册资金：<%=com.getEnrol_fund() %></td></tr>
        <tr><td>注册时间：<%=com.getEnrol_date() %></td></tr>
        <%
        Iterator it = com.getDepts().iterator();
        while (it.hasNext()) {
            Dept d = (Dept) it.next();
        %>
        <tr><td>部门名称：<%=d.getName() %></td></tr>                    //输出该公司对应的部门信息
        <%}
        h.getSession().clear();                                          //清空 Session 缓存
        %>
        <tr><td> </td></tr>
        <tr><td><div align="left"><font size="5" color="#FF3399"><b>从部门到公司</b></font></div></td></tr>
        <tr><td> </td></tr>
        <%
        Dept d = h.loadDept(1);                                          //查询指定 id 的部门对象
        com = d.getCompany();                                            //得到部门相对应的公司对象
        %>
        <tr><td>部门名称：<%=d.getName() %></td></tr>                    //打印基本信息
        <tr><td>公司名称：<%=com.getCompany_name() %></td></tr>
        <tr><td>公司类型：<%=com.getCompany_type() %></td></tr>
        <tr><td>公司法人：<%=com.getOwner() %></td></tr>
        <tr><td>注册资金：<%=com.getEnrol_fund() %></td></tr>
        <tr><td>注册时间：<%=com.getEnrol_date() %></td></tr>
        <%
            h.closeSession();
%>
```

秘笈心法

心法领悟 457：迫切左外连接查询的另一种实现。

在映射文件中配置迫切左外连接检索策略，只对 Session 的 load()方法和 get()方法有效，对 HQL 查询是不起作用的，如果在映射文件中采用延迟检索策略，但没有配置迫切左外连接检索策略，又需要立即访问关联对象，可以通过 HQL 查询语句实现迫切左外连接检索策略，将映射文件与 HQL 查询进行合理搭配，也是一种提升软件性能的手段。

17.3　Hibernate 集合映射与事务应用

实例 458	通过映射 Set 集合实现添加数据 光盘位置：光盘\MR\17\458	中级 实用指数：★★☆

实例说明

本实例实现的是通过映射 Set 集合添加公司信息。运行程序，将会看到添加新公司页面。填写信息后，单

击“提交”按钮，会转向另一个页面，在其中将会显示新添加公司的信息，如图 17.13 所示。

新添加公司信息

公司名称：开启梦想公司
公司类型：私营
公司法人：经理
注册资金：2450
注册时间：2002-09-12
部门名称：研发部
部门名称：运输部
部门名称：质量部
返回

图 17.13　通过映射 Set 集合实现添加数据

关键技术

要通过映射 Set 集合添加数据，需要在持久化类对应的映射文件中添加<set>元素。下面对<set>元素的属性及其子元素进行介绍。

- name 属性：在持久化类中添加的集合类型属性名。
- table 属性：关联表的名称，不配置的情况下 Hibernate 默认为与 name 属性值同名。
- cascade 属性：级联操作级别，建议在本实例中设置为 all。
- lazy 属性：是否采用延迟检索策略，在本实例中没有采用。
- <set>元素的子元素<key>：用来映射关联表的外键。其中，column 属性为关联表的外键字段名。
- <set>元素的子元素<element>：映射关联表的另一个字段。其中，column 属性为关联表的另一个字段名；type 属性为该字段的类型；not-null 属性用来设定该字段是否允许为空，建议设置为不允许为空。

设计过程

（1）创建公司信息类，主要是定义公司信息的一些属性。

（2）创建公司信息类的映射文件，具体代码如下：

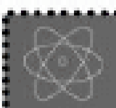
```
<class name="mrmwq.hibernate.Company" table="tb_company">
  <id name="id" column="id" type="int">                                              //定义表主键的映射
    <generator class="increment"/>
  </id>
  <property name="company_name" column="company_name" type="string" not-null="true"/>     //定义类属性
  <property name="company_type" column="company_type" type="string" not-null="true"/>
  <property name="owner" column="owner" type="string" not-null="true"/>
  <property name="enrol_fund" column="enrol_fund" type="int" not-null="true"/>
  <property name="enrol_date" column="enrol_date" type="date" not-null="true"/>
  <set name="depts" table="tb_dept2" cascade="all" lazy="false">                  //定义<set>元素
    <key column="company_id"/>
    <element column="name" type="string" not-null="true"/>                     //映射关联表中的另一个字段
  </set>
</class>
```

（3）创建操作数据库的 JSP 页面，同时把新添加的公司信息显示在页面上。具体代码如下：

```
<%
    Company com=(Company)h.queryOne("from Company where id in(select max(id) from Company)");    //查询新添加的公司对象
%>
      <tr><td>公司名称：<%=com.getCompany_name() %></td></tr>                    //显示公司信息
      <tr><td>公司类型：<%=com.getCompany_type() %></td></tr>
      <tr><td>公司法人：<%=com.getOwner() %></td></tr>
      <tr><td>注册资金：<%=com.getEnrol_fund() %></td></tr>
      <tr><td>注册时间：<%=com.getEnrol_date() %></td></tr>
      <%
      Iterator it=com.getDepts().iterator();
      while(it.hasNext()){
      %>
      <tr><td>部门名称：<%=it.next().toString() %></td></tr>
      <%
      }
      %>
```

心法领悟 458：Set 集合的说明。

Set 集合不对其中的对象进行排序，并且不能存放重复对象。Set 集合实现了 Set 接口，Set 接口继承于 Collection 接口。常用的 Set 接口实现类有 HashSet 类和 LinkedHashSet 类。



实例 459 通过映射 List 集合实现添加数据

光盘位置：光盘\MR\17\459

中级

实用指数：★★☆

实例说明

本实例实现的是通过映射 List 集合添加公司信息。运行程序，将会看到添加新公司页面。填写信息后，单击“提交”按钮，会转向另一个页面，其中将会显示新添加公司的信息，如图 17.14 所示。

新添加公司信息

公司名称：核心科技
公司类型：私营
公司法人：经理
注册资金：231
注册时间：2009-09-01
部门名称：拓展部
部门名称：政治部
部门名称：人事部
返回

图 17.14　通过映射 List 集合实现添加数据

关键技术

要通过映射 List 集合添加数据，需要在持久化类对应的映射文件中添加<list>元素。该元素和<set>元素的用法类似，对于其配置，只需要在其子元素<key>和<element>之间添加一个<index>元素，用来映射排序时参考的索引字段（<index>元素的 column 属性为字段名称），其他的配置信息及其含义不变。

（1）创建公司信息类，主要用于定义公司信息的一些属性。

（2）创建公司信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.Company" table="tb_company">
  <id name="id" column="id" type="int">                                                    //定义表主键的映射
    <generator class="increment"/>
  </id>
  <property name="company_name" column="company_name" type="string" not-null="true"/>     //定义类属性
  <property name="company_type" column="company_type" type="string" not-null="true"/>
  <property name="owner" column="owner" type="string" not-null="true"/>
  <property name="enrol_fund" column="enrol_fund" type="int" not-null="true"/>
  <property name="enrol_date" column="enrol_date" type="date" not-null="true"/>           //定义<list>元素
  <list name="depts" table="tb_dept3" cascade="all" lazy="false">
    <key column="company_id"/>
    <index column="locality"/>                                                             //映射排序时参考的索引字段
    <element column="name" type="string" not-null="true"/>                                 //映射关联表中的另一个字段
  </list>
</class>
```

（3）创建操作数据库的 JSP 页面，同时把新添加的公司信息显示在页面上。具体代码如下：

```
<%
    Company com=(Company)h.queryOne("from Company where id in(select max(id) from Company)");   //查询新添加的公司对象
 %>
        <tr><td>公司名称：<%=com.getCompany_name() %></td></tr>                                  //显示公司信息
        <tr><td>公司类型：<%=com.getCompany_type() %></td></tr>
        <tr><td>公司法人：<%=com.getOwner() %></td></tr>
        <tr><td>注册资金：<%=com.getEnrol_fund() %></td></tr>
        <tr><td>注册时间：<%=com.getEnrol_date() %></td></tr>
        <%
        Iterator it=com.getDepts().iterator();
        while(it.hasNext()){
        <%
        List list=com.getDepts();                                                                //得到部门对象的 List 集合
        for(int i=0;i<list.size();i++){
        %>
        <tr>
          <td>部门名称：<%=list.get(i)%></td>                                                    //循环输出部门名称信息
        </tr>
        <%}
          %>
```

秘笈心法

心法领悟 459：List 集合的说明。

List 集合的特点是将其中的对象按照索引进行排序，并且可以通过索引获取指定对象。另外，List 集合中可以存放重复对象，List 集合实现了 List 接口，List 接口继承于 Collection 接口。常用的 List 接口实现类有 ArrayList 类和 LinkedList 类。

实例 460 通过映射 Map 集合实现添加数据

光盘位置：光盘\MR\17\460

中级

实用指数：★★☆

实例说明

本实例实现的是通过映射 Map 集合添加公司信息。运行程序，将会看到添加新公司页面。填写信息后，单击“提交”按钮，会转向另一个页面，在其中将会显示新添加公司的信息，如图 17.15 所示。

新添加公司信息

公司名称：动力集团
公司类型：国营
公司法人：经理
注册资金：4500
注册时间：2010-09-10
部门名称：产业部
部门名称：评估部
部门名称：能源部
返回

图 17.15 通过映射 Map 集合实现添加数据

关键技术

要通过映射 Map 集合添加数据，需要在持久化类对应的映射文件中添加<map>元素。该元素和<list>元素的用法类似，对于其配置，只需要再配置<index>元素的 type 属性，同时指定 column 属性的类型即可，其他的配置信息及其含义不变。

（1）创建公司信息类，主要用于定义公司信息的一些属性。

（2）创建公司信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.Company" table="tb_company">
  <id name="id" column="id" type="int">                                                    //定义表主键的映射
    <generator class="increment"/>
  </id>
  <property name="company_name" column="company_name" type="string" not-null="true"/>      //定义类属性
  <property name="company_type" column="company_type" type="string" not-null="true"/>
  <property name="owner" column="owner" type="string" not-null="true"/>
  <property name="enrol_fund" column="enrol_fund" type="int" not-null="true"/>
  <property name="enrol_date" column="enrol_date" type="date" not-null="true"/>            //定义<ist>元素
  <map name="depts" table="tb_dept4" cascade="all" lazy="false">
    <key column="company_id"/>
    <index column="locality" type="string"/>                                                //映射排序时参考的索引字段
    <element column="name" type="string" not-null="true"/>                                  //映射关联表中的另一个字段
  </map>
</class>
```

（3）创建操作数据库的 JSP 页面，同时把新添加的公司信息显示在页面上。具体代码如下：

```
<%
    Company com=(Company)h.queryOne("from Company where id in(select max(id) from Company)");   //查询新添加的公司对象
%>
      <tr><td>公司名称：<%=com.getCompany_name() %></td></tr>                                     //显示公司信息
      <tr><td>公司类型：<%=com.getCompany_type() %></td></tr>
      <tr><td>公司法人：<%=com.getOwner() %></td></tr>
      <tr><td>注册资金：<%=com.getEnrol_fund() %></td></tr>
      <tr><td>注册时间：<%=com.getEnrol_date() %></td></tr>
      <%
      Iterator it=com.getDepts().iterator();
      while(it.hasNext()){
        <%
      Map map=com.getDepts();                                                                   //得到部门对象的 Map 集合
```

```
    %>
    <tr>
      <td>部门名称：<%=map.get("A")%></td>                                                    //循环输出部门名称信息
    </tr>
    <tr>
      <td>部门名称：<%=map.get("B")%></td>
    </tr>
    <tr>
      <td>部门名称：<%=map.get("C")%></td>
    </tr>
```

秘笈心法

心法领悟 460：Map 集合的说明。

Map 集合中的每个元素都是由一个键对象和一个值对象组成，即由一个(key,value)组成。其中，键对象（key）不可以重复，值对象（value）可以重复。

实例 461 事务回滚的应用

光盘位置：光盘\MR\17\461

中级

实用指数：★★☆

实例说明

本实例实现的是利用事务回滚来模拟银行转账的功能。运行程序，将会看到一个转账界面。选择“转出方”和“转入方”人员，然后填写转入金额。当转出方卡中金额大于输入的金额时，会提示转账成功，同时转出方的金额减少，转入方的金额增加；当转出方卡中金额小于输入的金额时，会提示转账失败，转出方和转入方的金额保持不变。本实例的运行结果如图 17.16 所示。

转账信息列表

已成功帮您转账！

转账前A用户的金额是:	134.0元
转账前B用户的金额是:	2.0元
转账后A用户的金额是:	94.0元
转账后B用户的金额是:	42.0元
你的转账金额是:	40元

转账信息列表

您的此次转账失败！

转账前A用户的金额是:	94.0元
转账前B用户的金额是:	42.0元
转账后A用户的金额是:	94.0元
转账后B用户的金额是:	42.0元
你的转账金额是:	200元

图 17.16　事务回滚的应用

关键技术

要实现提示转账成功或者失败信息，就要通过事务的提交或者回滚进行控制。事务具有如下特性。

- 原子性：每个事务都是一个不可分割的整体，只有所有的操作单元执行成功，整个事务才成功；否则此次事务将失败，所有执行成功的操作单元必须撤销，数据库回到此次事务之前的状态。
- 一致性：在执行一次事务后，关系数据的完整性和业务逻辑的一致性不能被破坏。
- 隔离性：在并发环境中，一个事务所做的修改必须与其他事务所做的修改相隔离。
- 持久性：事务结束后，对数据的修改是永久保存的，即使系统故障导致重启数据库系统，数据依然是修改后的状态。

（1）创建个人信息类，在该类中主要定义个人信息的一些属性。

（2）创建个人信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.Save" table="tb_save">
  <id name="id" column="id" type="int">                                                    //定义表主键的映射
    <generator class="increment"/>
  </id>
```

```
    <property name="name" column="name" type="string" not-null="true"/>                    //定义类的属性
    <property name="money" column="money" type="float" not-null="false"/>
</class>
```

（3）创建操作数据库的 JSP 页面，同时把转账后的详细信息显示在页面上。具体代码如下：

```
<%
    String smoney=request.getParameter("money");
    float money=Float.parseFloat(smoney);
    String from=new String(request.getParameter("from").getBytes("iso-8859-1"),"GBK");
    String to=new String(request.getParameter("to").getBytes("iso-8859-1"),"GBK");
    Save Aprice= h.QueryAprice(from);                                    //查询转账之前 A 对象
      Save Bprice=h.QueryBprice(to);                                     //查询转账之前 B 对象
    boolean transfer=h.testRollback(from,to,money);                      //转账操作
    String clew="您的此次转账失败！";
    if(transfer){
      clew="已成功帮您转账！";
    }
    Save Aprice2= h.QueryAprice(from);                                   //查询转账之后 A 对象
      Save Bprice2=h.QueryBprice(to);                                    //查询转账之后 B 对象
%>
  <tr align="center">                                                    //输出转账之前和之后的详细信息
      <td> </td>
      <td><font color="red"><%=clew %></font>
        <table width="385" border="1">
        <tr>
          <td width="187"><span class="STYLE1">转账前 A 用户的金额是：</span></td>
          <td width="182"><%=Aprice.getMoney()%>元</td>
        </tr>
        <tr>
          <td><span class="STYLE1">转账前 B 用户的金额是：</span></td>
          <td><%=Bprice.getMoney()%>元</td>
        </tr>
        <tr>
          <td><span class="STYLE1">转账后 A 用户的金额是：</span></td>
          <td><%=Aprice2.getMoney()%>元</td>
        </tr>
        <tr>
          <td><span class="STYLE1">转账后 B 用户的金额是：</span></td>
          <td><%=Bprice2.getMoney()%>元</td>
        </tr>
        <tr>
          <td><span class="STYLE1">你的转账金额是：</span></td>
          <td><%=smoney%>元</td>
          </tr>
      </table>
        </td>
      <td> </td>
  </tr>
```

秘笈心法

心法领悟 461：事务回滚的原因。

在本实例中，提示转账成功和失败信息是通过事务来控制的。当然，事先要在 tb_save 表中的 money 字段上建立约束条件，约束该字段的值必须大于 0。当该字段的值小于 0，此时再进行转账操作，事务就会回滚，然后显示转账失败信息，同时转账前后的金额保持不变。

实例 462	配置持久化类实现乐观锁的使用 光盘位置：光盘\MR\17\462	中级 实用指数：★★★

实例说明

本实例通过乐观锁来模拟银行转账系统。运行程序，在银行转账模拟系统主界面中可以看到有两个单选按

钮，当选中“模拟中间有其他人转账”单选按钮时，用户输入转账金额后将跳转到转账失败界面，同时 A 用户和 B 用户的账户金额保持不变；当选中“模拟中间无其他人转账”单选按钮时，用户输入转账金额后将跳转到转账成功界面，同时 A 用户的金额减少，B 用户的金额增加，如图 17.17 所示。

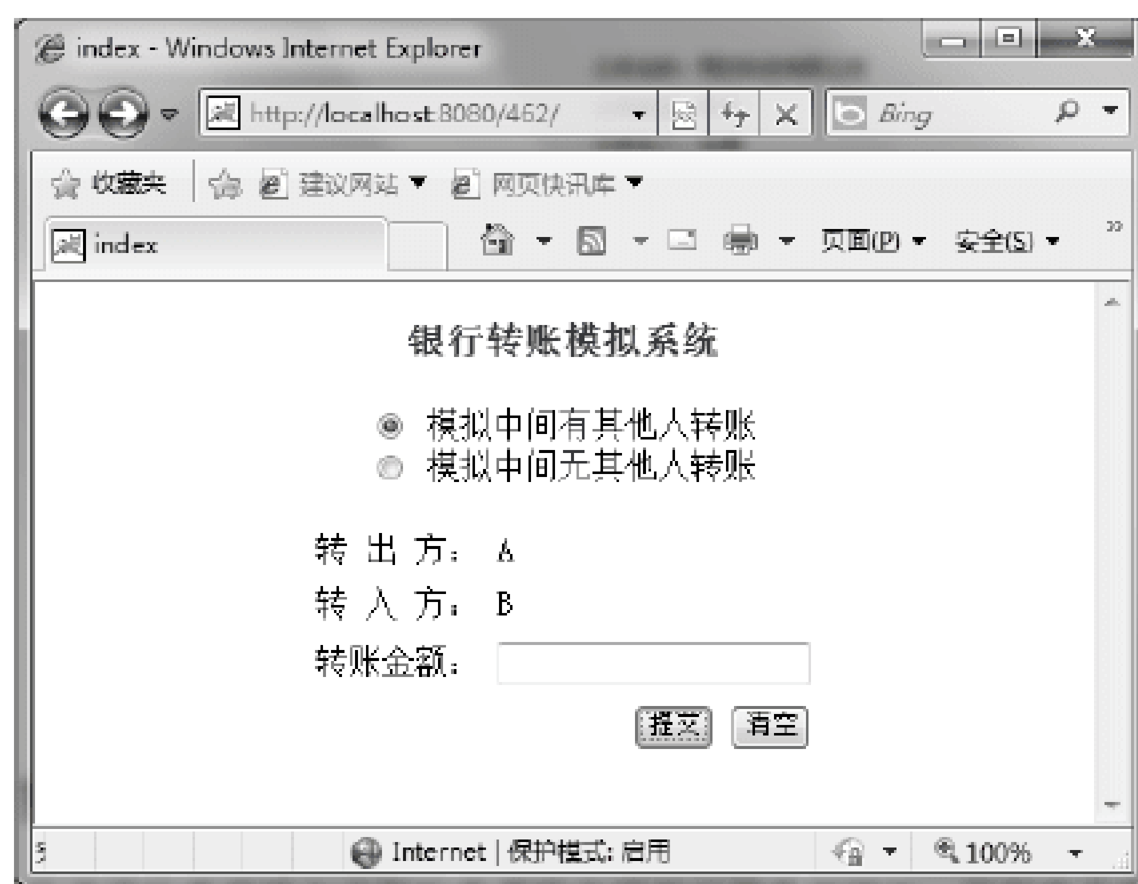


图 17.17　乐观锁的使用

关键技术

乐观锁就是指乐观地认为每个事务在操作数据时，很少有其他事务同时访问该数据资源，因而不在数据库层锁定，而是通过应用程序逻辑来实现版本控制。

Hibernate 提供了两种实现乐观锁的方法。

（1）方法一：通过配置持久化类的<version>元素。

在该类相对应的映射文件中，<class>元素的 optimistic-lock 属性的作用是指定乐观锁的实现策略，有 4 个可选值。

- none：不使用乐观锁。
- version：利用版本控制机制实现乐观锁。
- dirty：通过检查发生改变的属性实现乐观锁。
- all：通过检查所有的属性实现乐观锁。

Hibernate 官方推荐使用<version>元素实现乐观锁，该方式在对象脱离 Session 的情况下依然有效，其他锁机制则不具备这种特点。

（2）方法二：通过配置持久化类的<timestamp>元素。

该方法的实现机制与方法一基本相同，只是将版本信息字段由 int 型改为 Date 型，并赋予它实际的含义。每条记录的版本控制信息代表的是最后一次修改该记录的时间。它同样是由 Hibernate 自行控制，在访问一条记录的同时访问该记录的版本信息。如果对该记录的信息作了修改，在提交事务时，Hibernate 首先将访问时的版本信息与当前数据库中该记录的版本信息进行比对，如果一致则提交修改，并将原版本信息改为当前时间；如果不一致，则撤销此次修改。

注意：对于本实例，采用方法一进行说明。

（1）创建个人信息类，在该类中主要定义个人信息的一些属性。

（2）创建个人信息类的映射文件，具体代码如下：

```
<class name="mrmwq.hibernate.OptimisticLocking" table="tb_optimisticLockingV" optimistic-lock="version">    //指定乐观锁的实现策略
  <id name="id" column="id" type="int">                                                                    //定义表主键的映射
    <generator class="increment"/>
  </id>
  <version name="version" column="version" />                                                              //定义<version>元素
  <property name="name" column="name" type="string" not-null="true" />
  <property name="money" column="money" type="int" not-null="true" />
</class>
```

（3）创建操作数据库的 JSP 页面，同时把查询后的结果显示在页面上。具体代码如下：

```
<%
     List before=h.query("A","B");                                   //查询转账之前信息
     String test=request.getParameter("test");
     String smoney=request.getParameter("money");
     int money=Integer.parseInt(smoney);
     boolean transfer=h.update("A","B",money,test);                  //转账操作
     String clew="您此次转账失败！ ";
     if(transfer){
        clew="已成功帮您转账！ ";
     }
```

```
    List after=h.query("A","B");                                    //查询转账之后信息
    %>
  <tr align="center">
    <td> </td>
    <td><font color="red"><%=clew %></font></td>
    <td> </td>
  </tr>
  <tr>
    <td colspan="3"> </td>
  </tr>
  <tr align="center">
   <td> </td>
    <td>修改前的信息：</td>
    <td> </td>
  </tr>
  <%
  for(int i=0;i<before.size();i++){                                 //循环输出修改前信息
    OptimisticLocking b=(OptimisticLocking)before.get(i);
  %>
  <tr align="center">
    <td> </td>
    <td><%=b.getName()+":"+b.getMoney() %></td>
    <td> </td>
  </tr>
  <%
  }
  %>
  <tr>
    <td colspan="3"> </td>
  </tr>
  <tr align="center">
    <td> </td>
    <td>修改后的信息：</td>
    <td> </td>
  </tr>
  <%
  for(int i=0;i<after.size();i++){                                  //循环输出修改后信息
    OptimisticLocking b=(OptimisticLocking)after.get(i);
  %>
  <tr align="center">
    <td> </td>
    <td><%=b.getName()+":"+b.getMoney() %></td>
    <td> </td>
  </tr>
  <%
  }
  %>
  <tr>
    <td colspan="3"> </td>
  </tr>
  <tr>
    <td colspan="3"> </td>
    </tr>
```

心法领悟 462：方法一实现机制的说明。

该方法的实现机制是在数据库中添加一个 int 型的 version 字段，用来标记记录的版本信息。该字段由 Hibernate 自行控制，在访问一条记录的同时访问该记录的版本信息。如果对该记录的信息作了修改，在提交事务时，Hibernate 首先将访问时的版本信息与当前数据库中该记录的版本信息进行比对，如果一致则提交修改，并将原版本信息加 1；如果不一致，则撤销此次修改。

第18章

Spring 框架基础

- Spring 的依赖注入
- Spring 的事务管理
- Spring 的面向切面编程
- Spring 的持久化
- 在 Spring 中生成非 HTML 输出
- Spring 文件上传与国际化

18.1　Spring 的依赖注入

实例 463　应用 Setter 注入法实现 Bean 的注入

光盘位置：光盘\MR\18\463

高级

实用指数：★★★

实例说明

Spring 提供的 Setter 注入是最常用的一种 Bean 的注入法。本实例将实现这一功能，通过 Setter 注入后，直接将 Bean 的属性值输出在页面中，如图 18.1 所示。

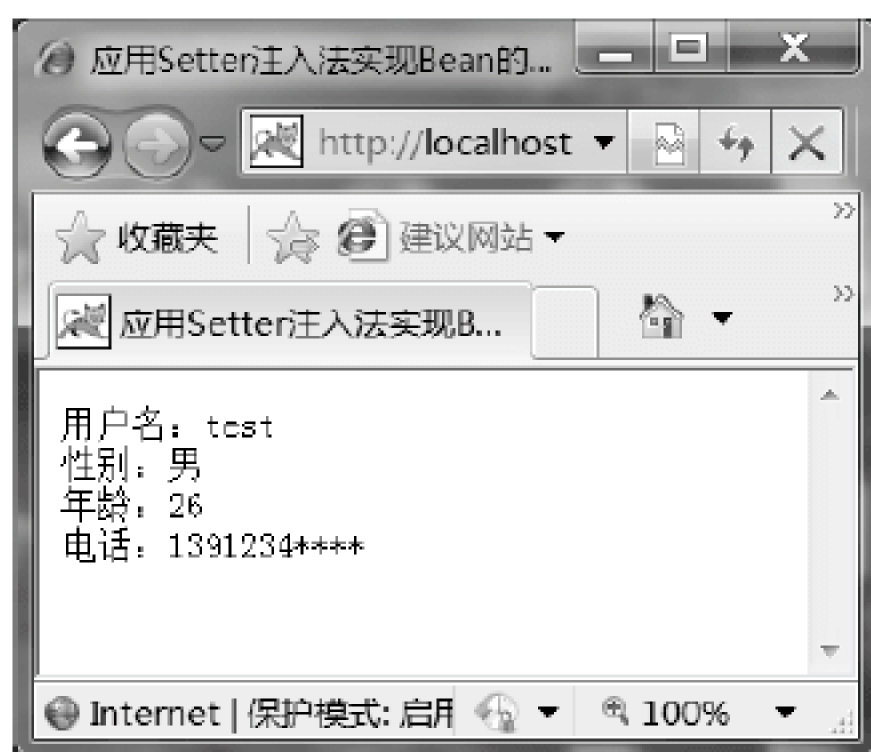


图 18.1　应用 Setter 注入后获取的 Bean 属性值

关键技术

实现 Bean 的 Setter 注入主要通过以下两个步骤。

（1）当前的 Bean 对象要注入到哪个对象中，就需要在相应对象中为当前对象设置私有属性，并设置属性的 getter 返回器和 setter 设置器。例如，在本实例中将 User 对象注入到 TestUtil 对象中，那么就需要在 TestUtil 类中设置 User 对象属性，并设置 getter 和 setter 方法。TestUtil 类的代码如下：

```
public class TestUtil {
        private User user;
        public User getUser() {
                return user;
        }
        public void setUser(User user) {
                this.user = user;
        }
}
```

（2）在 Spring 的 XML 配置文件中，通过<bean>标签对 Bean 进行依赖注入配置。<bean>标签的主要属性有两个：id 属性用于配置 Bean 的唯一标识，Bean 在注入时需要根据此标识查找 Bean 对象；class 属性用于指定完整的 Bean 类。代码如下：

```
<!-- 为 User 对象属性赋值 -->
<bean id="user" class="com.lh.entity.User" />
<!-- 配置 TestUtil，注入 User -->
<bean id="testUtil" class="com.lh.util.TestUtil">
        <property name="user">
                <ref local="user"/>
        </property>
</bean>
```

（1）创建 User 类（该类是一个普通的 JavaBean），设置几个属性并添加 getter 和 setter 方法。关键代码如下：

```
public class User {
        private String name = "test";
```

```
        private String sex = "男";
        private int age = 26;
        private String tel = "1391234****";
        ……//省略了属性的 getter 和 setter 方法
}
```

（2）创建 TestUtil 类，添加 User 属性，并设置 getter 和 setter 方法，然后编写 getUserInfo()方法。TestUtil 类的代码如下：

```
public class TestUtil {
 private User user;
 public User getUser() {
        return user;
 }

 public void setUser(User user) {
        this.user = user;
 }
 public boolean getUserInfo(){
        if(user!=null){
                return true;
        }else{
                return false;
        }

 }
}
```

（3）在 Spring 的 applicationContext.xml 中对 Bean 的依赖注入进行配置，具体代码参见关键技术部分。

（4）创建 index.jsp 页，首先装载 Spring 的配置文件，然后通过 Spring 的 BeanFactory 初始化 User 对象和 TestUtil 对象，并获取 User 对象的属性值。代码如下：

```
<%
        Resource cr = new ClassPathResource("applicationContext.xml");          //在 classpath 下读取指定的 XML 文件
        BeanFactory factory = new XmlBeanFactory(cr);                           //定义并建立 BeanFactory
        TestUtil testUtil = (TestUtil) factory.getBean("testUtil");             //指定 Bean 的名称来取得 Bean 实例
%>
<%if(testUtil.getUserInfo()){
        User user = testUtil.getUser();
%>
        用户名：<%=user.getName() %><br>
        性别：<%=user.getSex() %><br>
        年龄：<%=user.getAge() %><br>
        电话：<%=user.getTel() %><br>
<%} %>
```

秘笈心法

心法领悟 463：<property>子标签。

<bean>标签中包含一个<property>子标签，Spring 会根据<property>的配置实现依赖注入。其 name 属性值与 Bean 中设置的属性值对应；<ref>用于指定要注入的对象，其 local 属性值与配置文件中指定的<bean>的 id 属性值对应。

实例 464　应用构造器注入法实现 Bean 的注入

光盘位置：光盘\MR\18\464

高级

实用指数：★★★

本实例将通过 Spring 的构造器注入法实现 Bean 的依赖注入，并在页面中获取注入之后的 User 对象的属性值，如图 18.2 所示。

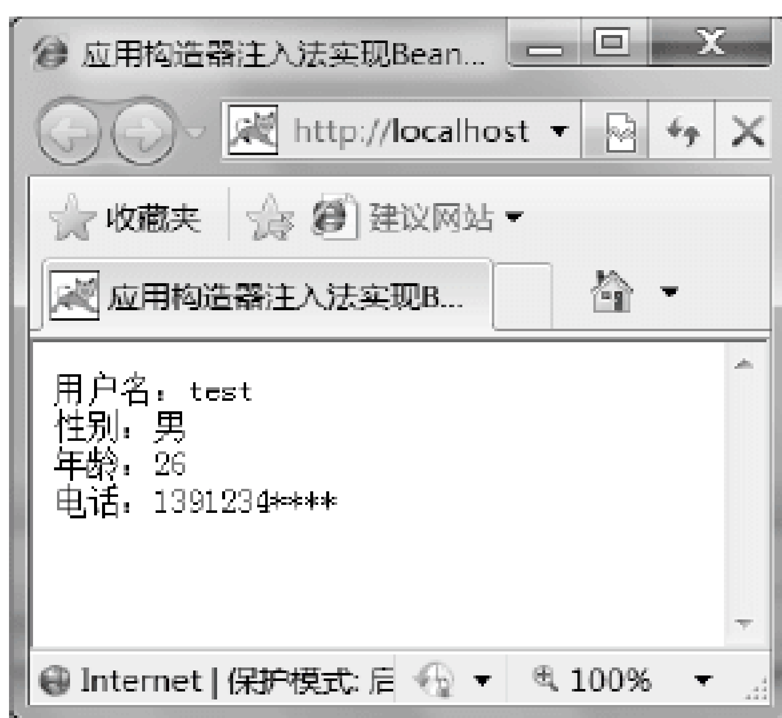


图 18.2　构造器注入法注入后获取的 Bean 属性值

关键技术

在类被实例化时，其构造方法将被调用并且只能调用一次。因此，构造器常被用于类的初始化操作。Spring 中的<constructor-arg>是<bean>元素的子元素，通过该元素可以实现为当前业务对象注入其所依赖的对象。<constructor-arg>元素中包含一个<ref>子元素，可使用<ref>引用容器中其他的对象实例。配置代码如下：

```
<!-- 为 User 对象属性赋值 -->
<bean id="user" class="com.lh.entity.User" />
<!-- 配置 TestUtil，注入 User -->
<bean id="testUtil" class="com.lh.util.TestUtil">
        <constructor-arg>
                <ref bean="user"/>
        <constructor-arg>
</bean>
```

设计过程

（1）本实例的实现是在实例 463 的基础上修改的，所以只需要在 TestUtil 类中创建一个构造方法，并传递一个 User 参数。代码如下：

```
public class TestUtil {
   private User user;
   public User getUser() {
         return user;
   }
   public void setUser(User user) {
         this.user = user;
   }
   public TestUtil(User user){//构造方法
         this.user = user;
   }
   public boolean getUserInfo(){
         if(user!=null){
                return true;
         }else{
                return false;
         }
   }
}
```

（2）在 Spring 的 XML 配置文件中应用<constructor-arg>元素通过构造器实现 Bean 的依赖注入，具体代码参见本实例的关键技术部分。

（3）创建 index.jsp 页，首先装载 Spring 的配置文件，然后通过 Spring 的 BeanFactory 初始化 User 对象和 TestUtil 对象，并获取 User 对象的属性值。具体代码参见配书光盘。

心法领悟 464：<constructor-arg>的 index 属性。

当某个业务对象的构造方法同时传入了多个参数时，可通过 index 属性指定参数的索引。index 属性的取值从 0 开始，所以指定第一个参数的 index 应该是 0，第二个参数的 index 应该是 1，以此类推。例如，TestUtil

类的构造方法有两个 String 类型的参数，那么<constructor-arg>元素的具体配置代码如下：

```
<bean id="testUtil" class="com.lh.util.TestUtil">
        <constructor-arg index="0" value="aaa" />
        <constructor-arg index="1" value="bbb" />
</bean>
```

实例 465 应用@Autowired 注解实现 Bean 的注入

光盘位置：光盘\MR\18\465

高级

实用指数：★★★

实例说明

可以利用 Spring 提供的@Autowired 注解实现 Bean 的注入。本例将在页面中输出通过@Autowired 注解注入的 Book 对象的属性值，如图 18.3 所示。

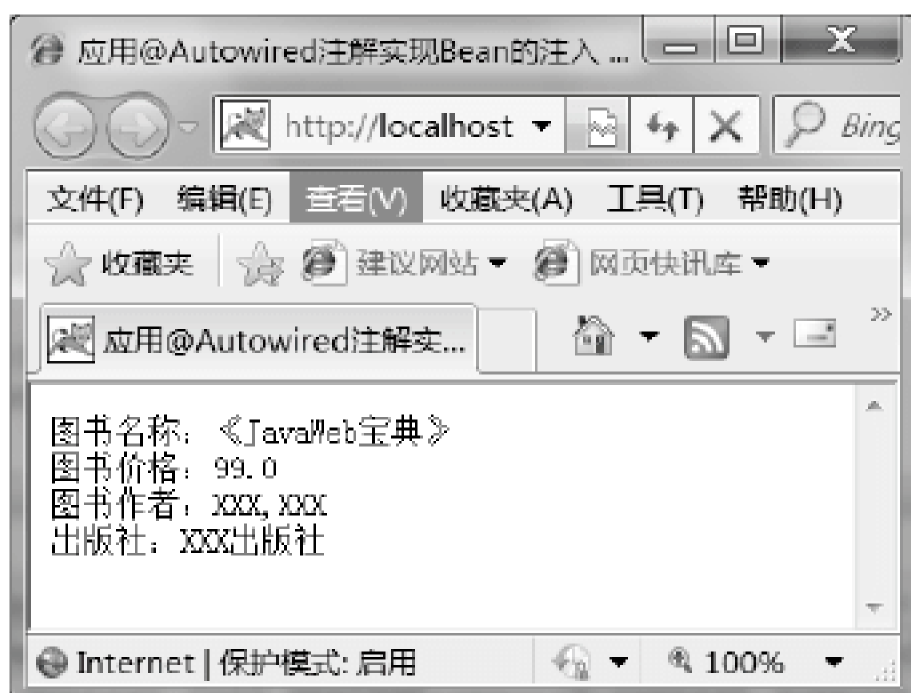


图 18.3　应用@Autowired 注解实现 Bean 的注入

通过@Autowired 注解实现依赖注入时，可以标注于以下 4 种不同的情况。

（1）可以在类的构造方法上标注@Autowired 注解，实现 Bean 的注入。例如，要将 User 对象注入到 Test 对象中，可以在 Test 类的构造方法中应用@Autowired 注解实现依赖注入。代码如下：

```
public class TestUtil {
private User user;            //声明要注入的对象

@Autowired
public TestUtil(User user){//构造方法
        this.user = user;
}
}
```

（2）将@Autowired 注解标注于属性，实现 Bean 的注入。在标注属性时，该属性需要提供 getter 和 setter 方法。示例代码如下：

```
public class TestUtil {

@Autowired
private User user;

public User getUser() {
        return user;
}
public void setUser(User user) {
        this.user = user;
}
}
```

（3）将@Autowired 注解标注于 setter 方法之上，实现 Bean 的注入。示例代码如下：

```
public class TestUtil {
private User user;
public User getUser() {
        return user;
}
@Autowired
public void setUser(User user) {
        this.user = user;
}
}
```

（4）将@Autowired 注解标注于任意方法之上（只要该方法定义了需要被注入的参数即可实现 Bean 的注

入）。示例代码如下：

```
public class TestUtil {
 private User user;

 @Autowired
 public void setMyUser(User user) {
          this.user = user;
 }
}
```

（1）创建 Book 类，代码如下：

```
public class Book {
 private String bookName = "《JavaWeb 宝典》";
 private double price = 99.00;
 private String author = "XXX,XXX";
 private String bookmaker = "XXX 出版社";
 ……
}
```

（2）创建 TestUtil 类，应用@Autowired 注解将 Book 对象进行注入。代码如下：

```
public class TestUtil {
 @Autowired
 private Book book;
 public Book getBook() {
          return book;
 }
 public void setBook(Book book) {
          this.book = book;
 }
 public boolean getBookInfo(){
          if(book!=null){
                   return true;
          }else{
                   return false;
          }
 }
}
```

（3）在 Spring 的配置文件中对 Bean 进行配置，并配置 AutowiredAnnotationBeanPostProcessor，该对象用于检查当前对象是否有@Autowired 标注的依赖需要注入。代码如下：

```
<beans
 xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">
 <!-- 配置 AutowiredAnnotationBeanPostProcessor 对象 ，用于检查@Autowired 标注需要注入的对象-->
 <bean class="org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor" />
 <!-- 配置 Book 对象 -->
 <bean id="book" class="com.lh.entity.Book" />
 <!-- 配置 TestUtil -->
 <bean id="testUtil" class="com.lh.util.TestUtil" />
</beans>
```

（4）创建 index.jsp 页面，通过 ApplicationContext 容器实现获取 Bean 对象。代码如下：

```
<%
      ApplicationContext context=new ClassPathXmlApplicationContext("applicationContext.xml");  //获取 ApplicationContext 容器
      TestUtil testUtil = (TestUtil) context.getBean("testUtil");                               //指定 Bean 的名称来取得 Bean 实例
%>
    <%if(testUtil.getBookInfo()){
      Book book = testUtil.getBook();
    %>
      图书名称：<%=book.getBookName() %><br>
      图书价格：<%=book.getPrice() %><br>
      图书作者：<%=book.getAuthor() %><br>
      出版社：<%=book.getBookmaker() %><br>
    <%} %>
```

秘笈心法

心法领悟 465：应用@Qualifier 注解对依赖注入的条件进行限制。

@Autowired 注解是按照类型进行匹配的，如果@Autowired 标注的依赖在容器中只能找到一个实例与之对应，那没有问题，但是如果存在多个同一类型的对象实例，@Qualifier 注解将起作用。假设 Test 类有两个实现，TestA 和 TestB，在 Spring 配置文件中的配置代码如下：

```
<beans>
<bean id="testA" class="……TestA" />
<bean id="testB" class="……TestB" />
<bean id="testUtil" class="……TestUtil" />
<beans>
```

如果想将 TestA 注入到 TestUtil 中，代码如下：

```
public class TestUtil {
        @Autowired
        @Qualifier("testA")
        private Test testA;
        private Test testB;
        ……
}
```

实例 466 应用@Resource 注解实现 Bean 的注入

光盘位置：光盘\MR\18\466

高级

实用指数：★★★

实例说明

除了可以使用 Spring 提供的@Autowired 注解标注相应类的定义之外，还可以使用 JSR 250 的@Resource 对相应类进行标注，同样可以达到依赖注入的目的。本例应用@Resource 注解实现 Bean 的注入后，在页面中输出了该对象的属性值，如图 18.4 所示。

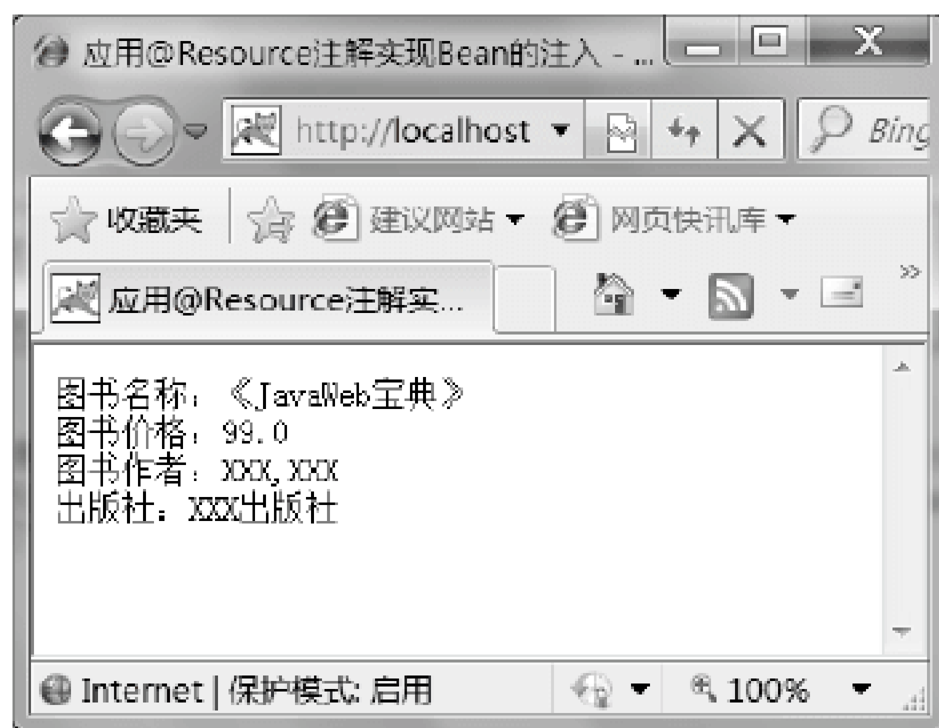


图 18.4　应用@Resource 注解实现依赖注入

关键技术

@Resource 与@Autowired 注解类似，除了可以直接在属性域上标注@Resource，还可以在构造方法或者普通方法定义上标注@Resource。如果@Resource 标注于属性域或者方法之上，相应的容器将负责把指定的资源注入给当前对象。

（1）本实例是在实例 465 的基础上修改的，因此首先需要在 TestUtil 类中将@Autowired 注解修改为@Resource 注解。代码如下：

```
public class TestUtil {
 @Resource(name="book") //指定要注入的对象，book 为 Spring 配置文件中定义的 Bean 的 id
 private Book book;
 public Book getBook() {
        return book;
 }
 public void setBook(Book book) {
        this.book = book;
 }
}
```

（2）在 Spring 的配置文件中对 Bean 进行配置。针对@Resource 注解，在配置文件中还需要配置 CommonAnnotationBeanPostProcessor，只有这样使用的@Resource 注解才能发挥作用。代码如下：

```
<beans>
<bean class="org.springframework.context.annotation.CommonAnnotationBeanPostProcessor" />
```

```
<!-- 配置 User 对象 -->
<bean id="book" class="com.lh.entity.Book" />
<!-- 配置 TestUtil -->
<bean id="testUtil" class="com.lh.util.TestUtil" />
<beans>
```

■ 秘笈心法

心法领悟 466：应用<context:annotation-config>元素。

不管是使用@Resource 或@Autowired 注解，都需要添加相应的 BeanPostProcessor 容器，但是如果使用<context:annotation-config>，就不必配置 BeanPostProcessor 了，因为<context:annotation-config>已经把 AutowiredAnnotationBeanPostProcessor 和 CommonAnnotationBeanPostProcessor 注册到容器了。在本实例中，如果应用<context:annotation-config>元素进行配置，则代码如下：

```
<beans
xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd">
<context:annotation-config />
<!--
<bean class="org.springframework.context.annotation.CommonAnnotationBeanPostProcessor" />
 -->
<!-- 配置 Book 对象 -->
<bean id="book" class="com.lh.entity.Book" />
<!-- 配置 TestUtil -->
<bean id="testUtil" class="com.lh.util.TestUtil" />
</beans>
```

实例 467　零配置实现 Bean 的注入

光盘位置：光盘\MR\18\467

高级　实用指数：★★★

■ 实例说明

在实例 465 和实例 466 中，无论是使用@Resource 还是@Autowired 注解，都需要在 Spring 的配置文件中对相应对象的 Bean 定义，区别就是没有在配置文件中明确指定依赖关系，而是应用了注解。但是这样还是不够彻底，此处想要的是不在配置文件中对 Bean 进行任何配置，只应用相应的注解实现依赖注入，也就是本实例所涉及的“零配置”。运行本实例，只应用注解实现依赖注入，并输出了注入对象的属性值，如图 18.5 所示。

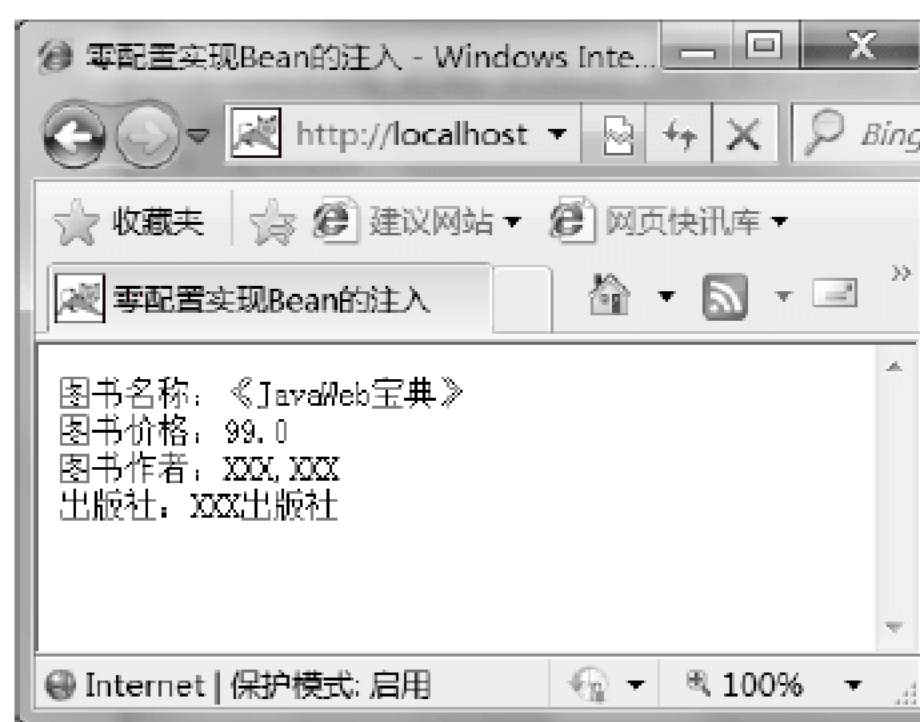


图 18.5　零配置实现 Bean 的注入

■

本例的实现主要用到了 classpath-scanning。当使用相应的注解对相关类进行标注后，classpath-scanning 会提取该类的相关信息，并添加到容器中，容器会根据注解的配置自动进行依赖注入。在 Spring 的配置文件中 classpath-scanning 功能需要由<context:component-scan>实现，所以在 Spring 的配置文件中添加<context:component-scan>后，classpath-scanning 功能就会开启。配置代码如下：

```
<beans
xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="
```

```
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd">
<context:component-scan base-package="com.lh" />
</beans>
```

配置完成后，<context:component-scan>会扫描 com.lh 路径下的所有标注了相应注解的类，并添加到 IOC 容器中，实现依赖注入。

<context:component-scan>默认扫描的注解类型是@Component，所以需要在相应的类中应用这个注解进行标注。TestUtil 类的代码如下：

```
@Component
public class TestUtil {
 @Autowired
 private Book book;
 public Book getBook() {
……
}
```

还需要对 Book 类标注@Component 注解，代码如下：

```
@Component
public class Book {
 private String bookName = "《JavaWeb 宝典》";
 private double price = 99.00;
 private String author = "XXX,XXX";
 private String bookmaker = "XXX 出版社";
……
}
```

设计过程

（1）创建 Book 类，并通过 Spring 的@Component 注解进行标注，因为需要将 Book 对象注入 TestUtil 中。具体代码参见本实例的关键技术部分。

（2）在 TestUtil 类中，同样需要应用@Component 注解进行标注，还需要应用@Autowired 注解标注要注入的对象的属性。具体代码参见本实例的关键技术部分。

（3）在 index.jsp 中，通过 ApplicationContext 获取相应的 Bean 对象。代码如下：

```
<%
    ApplicationContext context=new ClassPathXmlApplicationContext("applicationContext.xml");  //获取 ApplicationContext 容器
    TestUtil testUtil = (TestUtil) context.getBean("testUtil");                                //指定 Bean 的名称获取 Bean 实例
%>
  <%if(testUtil.getBookInfo()){
    Book book = testUtil.getBook();
  %>
    图书名称：<%=book.getBookName() %><br>
    图书价格：<%=book.getPrice() %><br>
    图书作者：<%=book.getAuthor() %><br>
    出版社：<%=book.getBookmaker() %><br>
  <%} %>
```

心法领悟 467：应用@Component 注解定义 Bean 的名称。

<context:component-scan>在扫描相关类定义并添加到容器时，会使用一个默认的命名规则生成需要添加到容器的 Bean 定义的名称。例如，在本实例中，TestUtil 通过默认的命名规则将获取 testUtil 作为 Bean 定义名称。如果需要改变这一默认行为，通过@Component 注解定义即可。代码如下：

```
@Component("test")
public class TestUtil {
 @Autowired
 private Book book;
 public Book getBook() {
……
}
```

实例 468　为 JavaBean 的集合对象注入属性值

光盘位置：光盘\MR\18\468

高级　实用指数：★★★

实例说明

在 Spring 中可以对 List、Set、Map 等集合进行配置，不过根据集合类型的不同，需要使用不同的标签配置对应的集合。本实例将实现这一功能，运行结果如图 18.6 所示，输出了注入集合后的元素值。

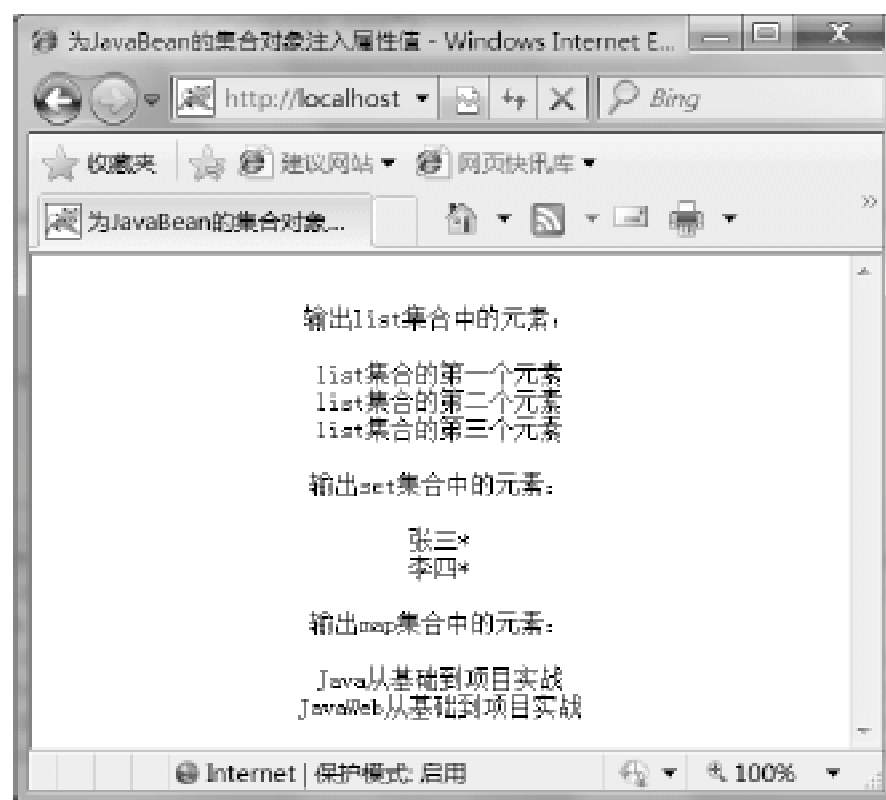


图 18.6　输出集合中的元素值

关键技术

在 Spring 的配置文件中，包含以下 3 种配置集合类型的元素。

（1）<list>元素

可通过<list>元素配置相应的 List 集合，然后通过<value>为 List 集合添加元素。通过<list>可以注入有序的依赖。示例代码如下：

```
<bean id="simple" class="Simple">
<property name="list">
        <list>
                <value>list 集合的第一个元素</value>
                <value>list 集合的第二个元素</value>
                <value>list 集合的第三个元素</value>
        </list>
</property>
</bean>
```

（2）<set>元素

<set>与<list>类似，只不过<set>添加的元素是无序的。示例代码如下：

```
<bean id="simple" class="Simple">
<property name="set">
        <set>
                <value>张三</value>
                <value>李四</value>
        </set>
</property>
</bean>
```

（3）<map>元素

与列表（list）使用数字下标来标识元素不同，<map>可以通过指定的键（key）获取相应的值。示例代码如下：

```
<bean id="simple" class="Simple">
<property name="map">
        <map>
                <entry key="key1" value="Java 从基础到项目实战"/>
                <entry key="key2" value="JavaWeb 从基础到项目实战"/>
        </map>
</property>
</bean>
```

（1）创建 TestUtil 类，在该类中定义 List、Set、Map 类型的属性，并设置 getter 和 setter 方法。代码如下：

```
public class TestUtil {
private List list;
private Set set;
private Map map;
public void setList(List list) {
        this.list = list;
}
public void setSet(Set set) {
```

```
        this.set = set;
}
public void setMap(Map map) {
        this.map = map;
}
public List getList() {
        return list;
}
public Set getSet() {
        return set;
}
public Map getMap() {
        return map;
}
}
```

（2）在 Spring 的配置文件中对 TestUtil 进行配置，并通过<list>、<set>、<map>为 TestUtil 的 List、Set、Map 集合属性赋值。代码如下：

```
<bean id="testUtil" class="com.lh.util.TestUtil">
        <property name="list">
                <list>
                        <value>list 集合的第一个元素</value>
                        <value>list 集合的第二个元素</value>
                        <value>list 集合的第三个元素</value>
                </list>
        </property>
        <property name="set">
                <set>
                        <value>张三*</value>
                        <value>李四*</value>
                </set>
        </property>
        <property name="map">
                <map>
                        <entry key="key1" value="Java 从基础到项目实战"/>
                        <entry key="key2" value="JavaWeb 从基础到项目实战"/>
                </map>
        </property>
</bean>
```

（3）在 index.jsp 中，读取集合中的元素值并输出。代码如下：

```
<%
        ApplicationContext context=new ClassPathXmlApplicationContext("applicationContext.xml");  //获取 ApplicationContext 容器
        TestUtil testUtil = (TestUtil) context.getBean("testUtil");                                //指定 Bean 的名称取得 Bean 实例
%>
    <center>
    <br>输出 list 集合中的元素：<br><br>
    <c:forEach var="item" items="<%=testUtil.getList() %>">
                ${item}<br/>
    </c:forEach>
   <br> 输出 set 集合中的元素：<br><br>
    <c:forEach var="setItem" items="<%=testUtil.getSet() %>">
                ${setItem}<br/>
    </c:forEach>
   <br> 输出 map 集合中的元素：<br><br>
    <c:forEach var="mapItem" items="<%=testUtil.getMap() %>">
                ${mapItem.value}<br/>
    </c:forEach>
    </center>
```

心法领悟 468：向集合中添加对象类型的元素。

<list>、< set>、<map>不仅可以添加 String 类型的元素，而且可以添加对象类型的元素。示例代码如下：

```
<bean id="book" class="com.lh.entity.Book" />
<bean id="testUtil" class="com.lh.util.TestUtil">
        <property name="list">
                <list>
```

```
                        <value>list 集合的第一个元素</value>
                        <value>list 集合的第二个元素</value>
                        <value>list 集合的第三个元素</value>
                        <ref bean="book"/>
                </list>
        </property>
        <property name="set">
                <set>
                        <value>张三*</value>
                        <value>李四*</value>
                        <ref bean="book" />
                </set>
        </property>
        <property name="map">
                <map>
                        <entry key="key1" value="Java 从基础到项目实战"/>
                        <entry key="key2" value="JavaWeb 从基础到项目实战"/>
                        <entry key="book">
                                <ref bean="book"/>
                        </entry>
                </map>
        </property>
</bean>
```

实例 469　使用<prop>标签为 Java 持久属性集注入值

高级

光盘位置：光盘\MR\18\469

实用指数：★★★

实例说明

本实例将使用 Spring 提供的<prop>为 Java 持久化属性集注入值，也就是向 java.util.Properties 对象中注入值。运行本实例，将输出 Properties 中的各个属性值，如图 18.7 所示。

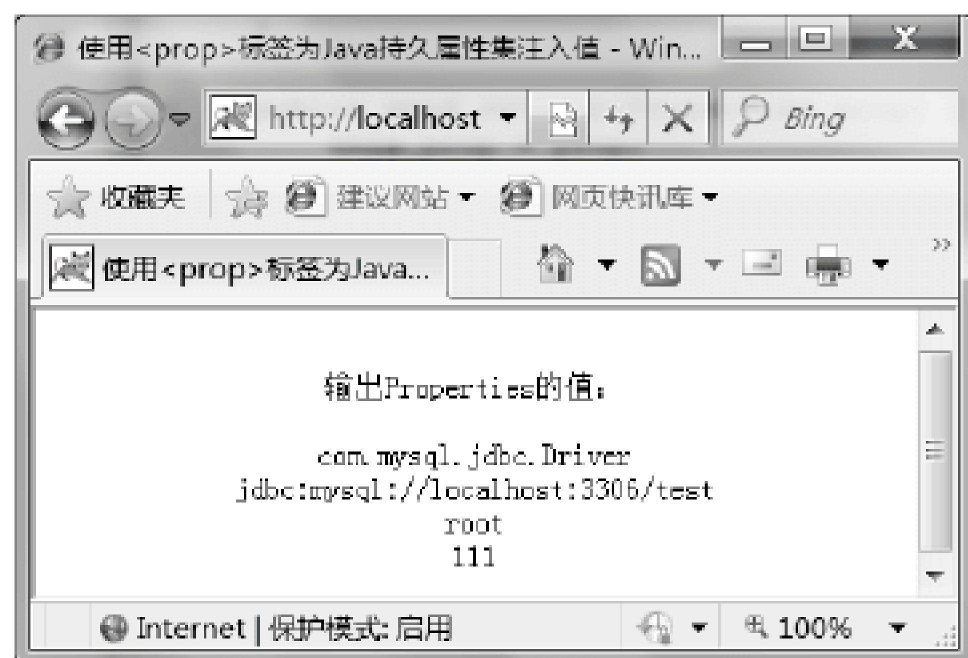


图 18.7　输出 Properties 中的各个属性值

关键技术

<props>是简化了的<map>，该元素对应配置类型为 java.util. Properties 的对象依赖。因为 Properties 只能指定 String 类型的键（key）和值，所以<props>的配置简化很多，只有固定的格式。示例代码如下：

```
<bean id="testUtil" class="com.lh.util.TestUtil">
        <property name="prop">
                <props>
                        <prop key="key1" /> Java 从基础到项目实战</prop>
                        <prop key="key2" /> JavaWeb 从基础到项目实战</prop>
                </map>
        </property>
</bean>
```

设计过程

（1）创建 TestUtil 类，并添加一个 Properties 类型的属性，然后添加 getter 和 setter 方法。代码如下：

```
public class TestUtil {
private Properties prop;
public Properties getProp() {
        return prop;
}
public void setProp(Properties prop) {
        this.prop = prop;
}
}
```

（2）编写 Spring 的配置文件，对 TestUtil 进行配置，并对 Properties 属性赋值。代码如下：

```
<bean id="testUtil" class="com.lh.util.TestUtil">
        <property name="prop">
                <props>
                        <prop key="driver">com.mysql.jdbc.Driver</prop>
                        <prop key="url">jdbc:mysql://localhost:3306/test</prop>
                        <prop key="username">root</prop>
                        <prop key="password">111</prop>
                </props>
        </property>
</bean>
```

（3）在 index.jsp 页中，获取 TestUtil 对象，然后输出 Properties 属性的各个值。具体代码参见配书光盘。

秘笈心法

心法领悟 469：<props>元素。

每个<props>可以嵌套多个<prop>，<prop>内部没有任何元素可以使用，只能是字符串，这是由 java.util.Properties 的语义决定的。

实例 470　按照 Bean 的名称自动装配 User

光盘位置：光盘\MR\18\470

高级

实用指数：★★★

实例说明

本实例将介绍如何按照 Bean 的名称自动装配 User 对象，程序运行结果如图 18.8 所示，在页面中输出了按照 Bean 名称自动装配的 User 对象的属性值。

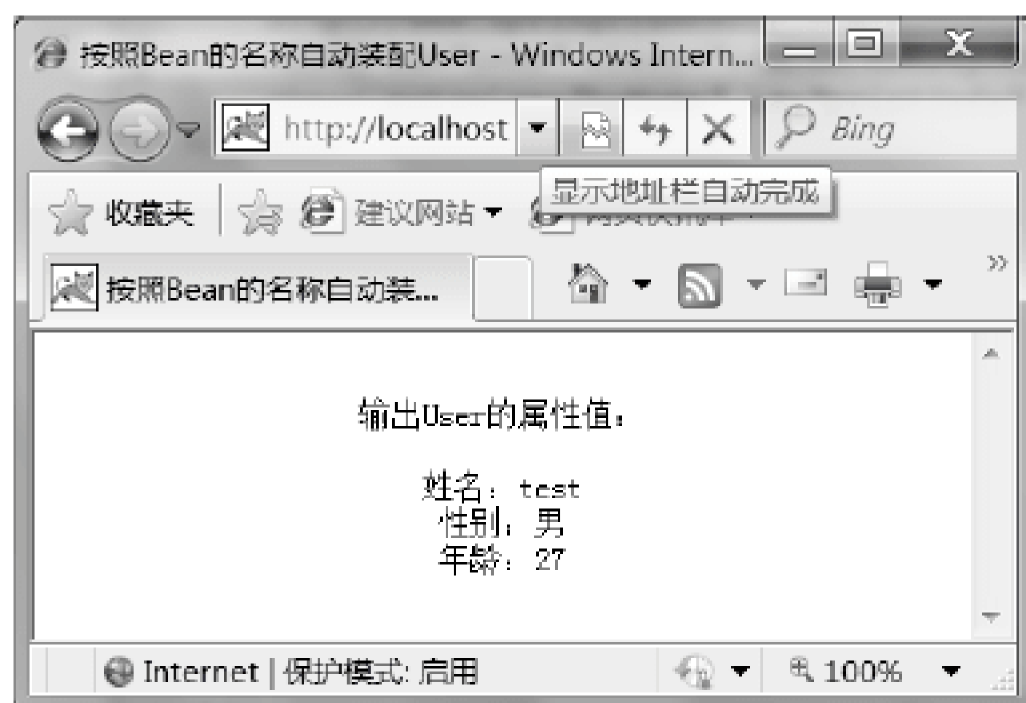


图 18.8　输出按照 Bean 名称自动装配的 User 对象的属性值

关键技术

<bean>元素的 autowire 属性负责自动装配<bean>标签，定义 JavaBean 的属性。这样做可以省去很多配置 JavaBean 属性的标签代码，使代码更整洁、美观；但是也有负面影响，使用自动装配之后，无法从配置文件中读懂 JavaBean 需要什么属性。

（1）User 对象中的关键代码如下：

```
public class User {
      private String name;              //用户姓名
      private Integer age;              //年龄
      private String sex;               //性别
      ……                                //省略的 setter 和 getter 方法
}
```

（2）定义 Bean（PrintInfo），将 User 对象注入到 PrintInfo 对象中。代码如下：

```
public class PrintInfo {
      private User user;                //注入 User 对象
      public User getUser() {
            return user;
      }
      public void setUser(User user) {
            this.user = user;
      }
}
```

（3）在 Spring 的配置文件 applicationContext.xml 中设置 Bean 的自动装配，Spring 将根据 Bean 中的属性名称自动将 User 对象注入到指定的 Bean 中。关键代码如下：

```
<bean id="user" class="com.lh.entity.User" />
<bean autowire="byName" id="printInfo" class="com.lh.util.PrintInfo" />
```

秘笈心法

心法领悟 470：按 Bean 名称自动装配存在的问题。

按 Bean 名称自动装配存在错误装配 JavaBean 的可能，如果配置文件中定义了与需要自动装配的 JavaBean 的名称相同而类型不同的 JavaBean，则会错误地注入不同类型的 JavaBean。

实例 471　按照 Bean 的类型自动装配 User

光盘位置：光盘\MR\18\471

高级

实用指数：★★★★

实例说明

本实例将演示如何按照 Bean 的类型自动装配 User 对象，程序运行结果如图 18.9 所示，在页面中输出了按照 Bean 的类型自动装配的 User 对象的属性值。

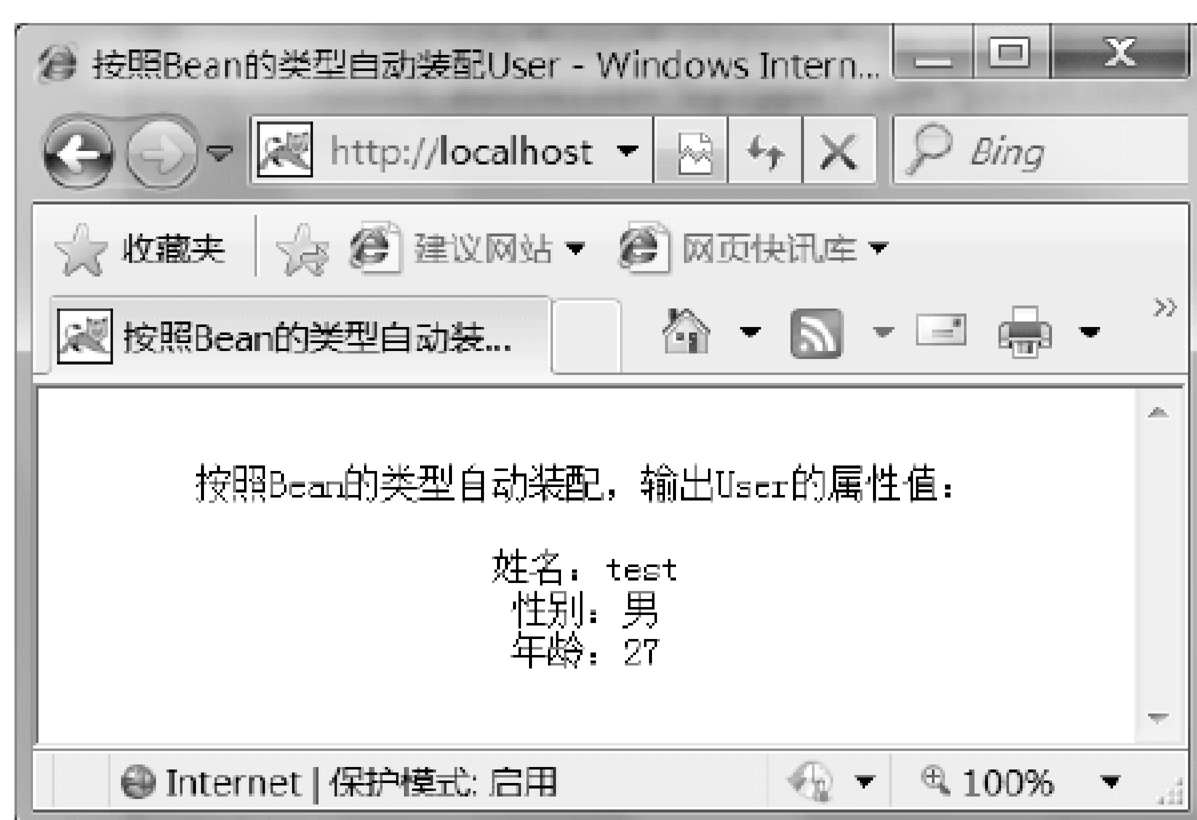


图 18.9　输出按照 Bean 类型自动装配的 User 对象的属性值

关键技术

Spring 以 Bean 类型区分自动装配，这次容器匹配的不再是 Bean 的名称，它会自动寻找与 JavaBean 的属性类型相同的 JavaBean 的定义，并将其注入到需要自动装配的 JavaBean 中。

设计过程

在实例 470 的基础上进行修改，将<bean>的 autowire 属性的 byName 改为 byType，其他保持不变。代码如下：

```
<bean id="user" class="com.lh.entity.User" />
<bean autowire="byType" id="printInfo" class="com.lh.util.PrintInfo" />
```

秘笈心法

心法领悟 471：按照 Bean 的类型自动装配的问题。

按照 Bean 的类型，自动装配也会出现无法自动装配的情况。例如，在配置文件中再次添加一个 User 类的实现对象，byType 自动装配类型会因为无法自动识别装配哪一个 JavaBean 而抛出 org.springframework.beans.factory. UnsatisfiedDependencyException 异常。要解决此问题，只能通过混合使用手动装配来指定装配哪个 JavaBean。

实例 472　配置 Bean 的延迟初始化

光盘位置：光盘\MR\18\472

高级

实用指数：★★★

实例说明

本实例将演示如何配置 Bean 的延迟初始化，程序运行结果如图 18.10 所示，在页面中输出了通过延迟初始

化配置的 User 对象的属性值。

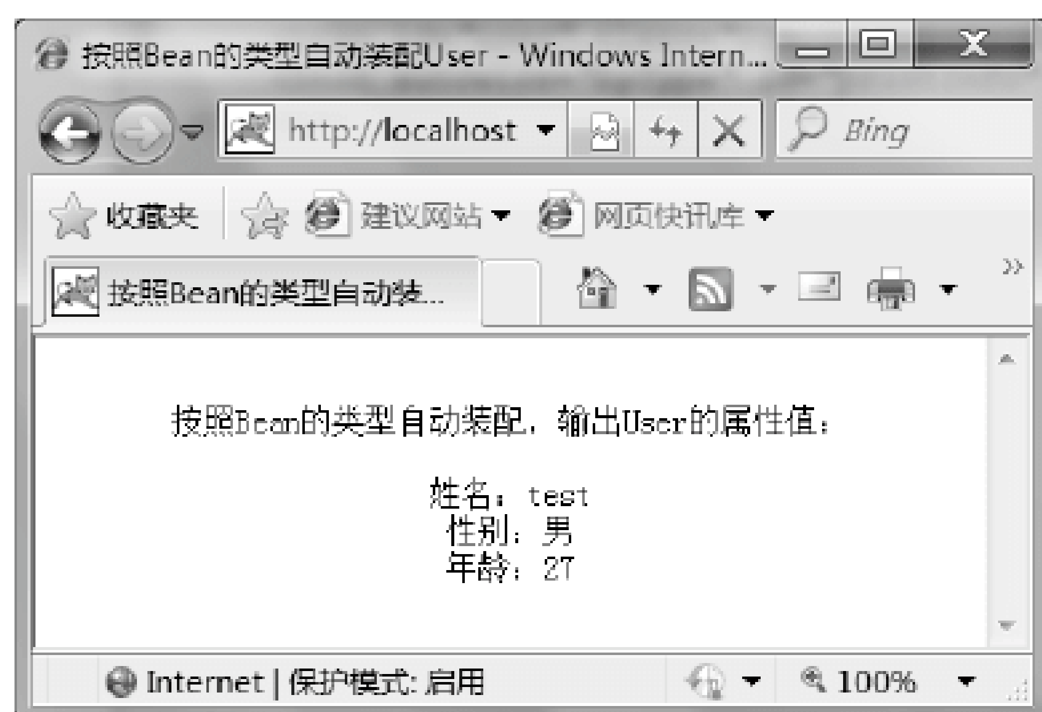


图 18.10　延迟初始化配置的 User 对象

关键技术

在配置<bean>时，可以通过 lazy-init 属性配置延迟初始化 Bean 对象。该特性主要是针对 ApplicationContext 容器的 Bean 初始化行为，ApplicationContext 在容器启动时，就会立即对所有的 Bean 定义进行实例化操作。默认的<bean>的 lazy-init 属性值为 false，只有当 laze-int 属性值为 true 时才会开启延迟初始化。

设计过程

在 Spring 的配置文件中，通过<bean>的 lazy-init 属性对 User 和 PrintInfo 设置延迟初始化。代码如下：

```
<bean id="user" class="com.lh.entity.User" lazy-init="true"/>
<bean id="printInfo" class="com.lh.util.PrintInfo" lazy-init="true">
        <property name="user">
                <ref bean="user"/>
        </property>
</bean>
```

秘笈心法

心法领悟 472：延迟初始化问题。

如果某个非延迟初始化的 Bean 依赖于通过 lazy-init 属性设置了延迟初始化的 Bean，那么会出现延迟初始化失败的问题。例如，User 对象设置了延迟初始化，而 PrintInfo 对象没有设置延迟初始化，并且 PrintInfo 对象依赖于 User 对象，那么容器还是会首先实例化 User，然后再实例化 PringInfo。这种互相牵连会导致延迟初始化失败。

实例 473　通过<beans>设置统一的延迟初始化行为

光盘位置：光盘\MR\18\473　　高级　　实用指数：★★★

实例说明

如果存在很多需要延迟初始化的 Bean，那么就需要为每个 Bean 设置延迟初始化，这样显得相当麻烦，并且由于依赖关系延迟初始化会出现问题。针对这一问题，可以通过<beans>设置统一延迟初始化策略，而不必为每个 Bean 设置延迟初始化。本例运行结果如图 18.11 所示，在页面中输出了通过<beans>设置统一延迟初始化的 User 对象属性值。

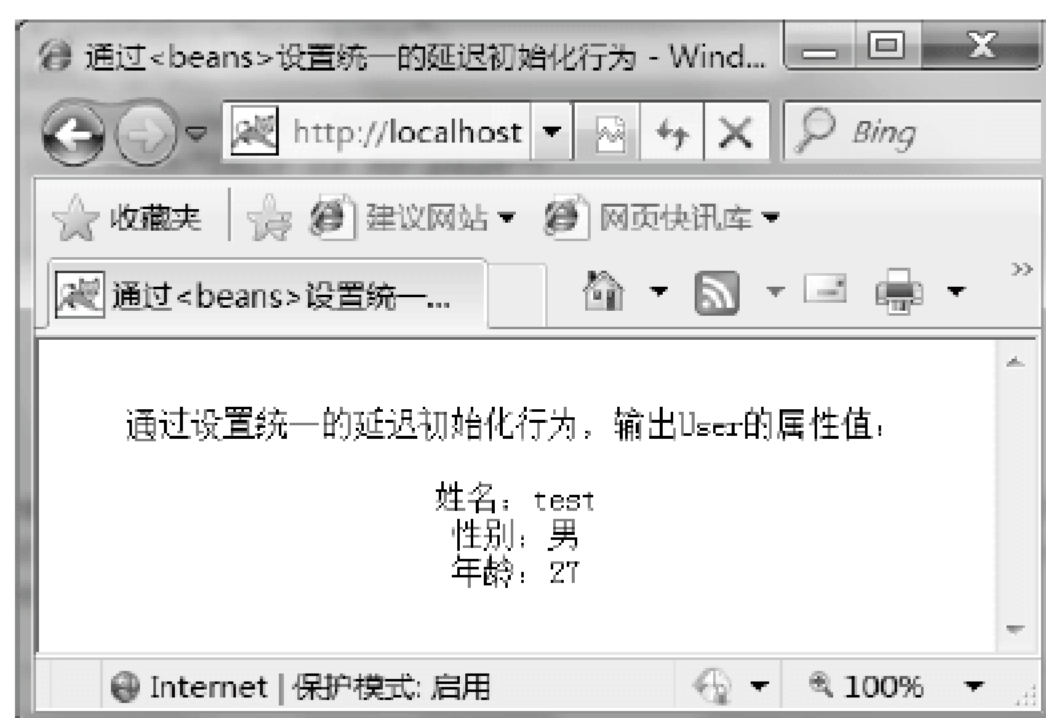


图 18.11　通过<beans>设置统一延迟初始化

关键技术

在<beans>元素中包含一个 default-lazy-init 属性，该属性值默认为 false；如果希望所有的 Bean 延迟初始化，可以设置 default-lazy-init 属性值为 true，这样就开启了所有 Bean 的延迟初始化操作。

在 Spring 的配置文件中，在<beans>中设置 default-lazy-init 属性值为 true，开启所有 bean 的延迟初始化操作。代码如下：

```
<beans default-lazy-init="true">
<bean id="user" class="com.lh.entity.User" />
```

```
<bean id="printInfo" class="com.lh.util.PrintInfo" >
        <property name="user">
                <ref bean="user"/>
        </property>
</bean>
</beans>
```

心法领悟 473：<beans>的 default-autowire 属性。

使用 default-autowire 属性可以设置所有<bean>定义的自动绑定。其默认值为 no，即不进行自动绑定。可以将 default-autowire 属性设置为 byName 或 byType，也就是根据 Bean 的名称或者类型实现自动绑定，这样可以省去为多个<bean>单独设置 autowire 属性的麻烦。

实例 474　自定义 MyDateEditor 编辑器实现类型转换　高级

光盘位置：光盘\MR\18\474　　实用指数：★★★

实例说明

本实例将自定义 MyDateEditor 属性编辑器，将 String 类型转换为 Date 类型，并输出获得的用户信息，如图 18.12 所示。

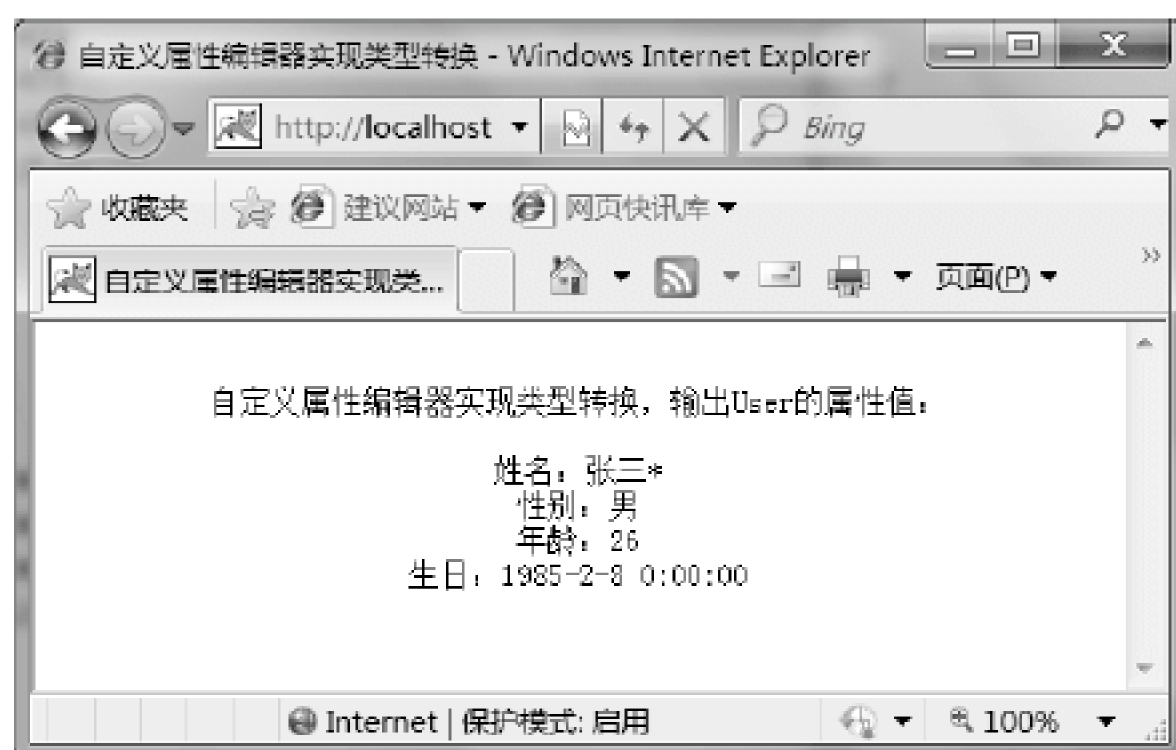


图 18.12　自定义属性编辑器运行结果

关键技术

属性编辑器来自于 java.beans.PropertyEditor 接口，支持各种不同类型显示和更新属性值的方式。大多数属性编辑器只需要支持 PropertyEditor 接口中的部分方法，而一些简单的属性编辑器可能只支持 getAsText()和 setAsText()方法。当定制与参数对象类型相对应的属性编辑器时，每个属性编辑器都必须编写 setValue()方法，每个属性编辑器都应该有一个空的构造方法。

PropertyEditor 接口有很多当前项目用不到的方法，如果用 PropertyEditor 接口定制编辑器，则需要实现接口中定义的所有方法。但是通过继承接口的实现类 java.beans.PropertyEditorSupport 定制属性编辑器，只需重写需要的方法（例如，setAsText()）即可，Spring 需要做的只是将现有的属性编辑器注册给指定的类。

设计过程

（1）建立 User 类来存储用户信息，其中除 3 个基本类型的姓名、年龄和性别属性之外，还有 java.util.Date 类型的出生日期属性。关键代码如下：

```
public class UserInfo {
    private String name;                                  //姓名
    private char sex;                                     //性别
    private int age;                                      //年龄
    private Date birthday;                                //出生日期
    ……                                                    //省略的 setter 和 getter 方法
}
```

（2）编写 Date 类型的属性编辑器 MyDateEditor 类，继承 PropertyEditorSupport 类（编写属性编辑器只需重写需要的方法，本实例定制的属性编辑器重写了 setAsText()方法），然后根据 String 类型的参数创建 Date 类型的对象。程序代码如下：

```
public class MyDateEditor extends PropertyEditorSupport {
    //设置日期格式方法
    public void setAsText(String text) throws IllegalArgumentException {
        Date date = new Date(text);
```

```
            setValue(date);                                         //设置日期格式
        }
}
```

（3）编写 Spring 的 JavaBean 配置文件 applicationContext.xml，其中定义了用户信息的 User 类。最主要的是注册自定义的属性编辑器 MyDateEditor 类，这样在给 User 类注入 Date 类型的出生日期属性时，可以直接输入日期字符串而不用再实例化 Date 对象。程序代码如下：

```
<bean id="user" class="com.lh.entity.User" >
        <property name="name">
                <value>张三*</value>
        </property>
        <property name="age">
                <value>26</value>
        </property>
        <property name="sex">
                <value>男</value>
        </property>
        <property name="birthday">
                <value>1985/2/8</value>
        </property>
</bean>
<bean id="printInfo" class="com.lh.util.PrintInfo" >
        <property name="user">
                <ref bean="user"/>
        </property>
</bean>
<bean id="customEditorConfigurer"
class="org.springframework.beans.factory.config.CustomEditorConfigurer">
        <property name="customEditors">
                <map>
                        <entry key="java.util.Date">
                                <bean id="MyDateEditor" class="com.lh.util.MyDateEditor" />
                        </entry>
                </map>
        </property>
</bean>
```

（4）创建 index.jsp 页，应用 ApplicationContext 获取 Bean 对象，然后输出属性值。代码如下：

```
<%
        ApplicationContext context=new ClassPathXmlApplicationContext("applicationContext.xml");  //获取 ApplicationContext 容器
        PrintInfo printInfo = (PrintInfo) context.getBean("printInfo");                          //指定 Bean 的名称来取得 Bean 实例
        User user = printInfo.getUser();
%>
    <center>
    <br>自定义属性编辑器实现类型转换，输出 User 的属性值：<br><br>
        姓名：<%=user.getName()%><br>
        性别：<%=user.getSex()%><br>
        年龄：<%=user.getAge()%><br>
        生日：<%=user.getBirthday().toLocaleString()%><br>
```

秘笈心法

心法领悟 474：Spring 的内置属性编辑器。

Spring 中有很多内置的属性编辑器，可以满足大部分需求，并且部分编辑器已经自动注册到容器中，程序可以直接使用这些编辑器，就像它们根本不存在一样。Spring 的属性编辑器包含在 org.springframework.beans.propertyeditors 包中。

实例 475	验证用户登录 光盘位置：光盘\MR\18\475	高级 实用指数：★★★

实例说明

本实例通过 Setter 注入的方式限定合法用户的用户名和密码，利用 Spring 的 IoC 技术实现用户登录的验证

机制，对用户进行登录验证。首先利用 Spring 的自动装配模式将 User 对象注入到控制器中，然后将用户输入的用户名和密码与系统中限定的合法用户的用户名和密码进行匹配。程序运行结果如图 18.13 所示，当用户名与密码匹配时跳转到登录成功页面，如图 18.14 所示；当用户名与密码不匹配时将跳转到登录失败页面，如图 18.15 所示。



图 18.13　用户登录页面

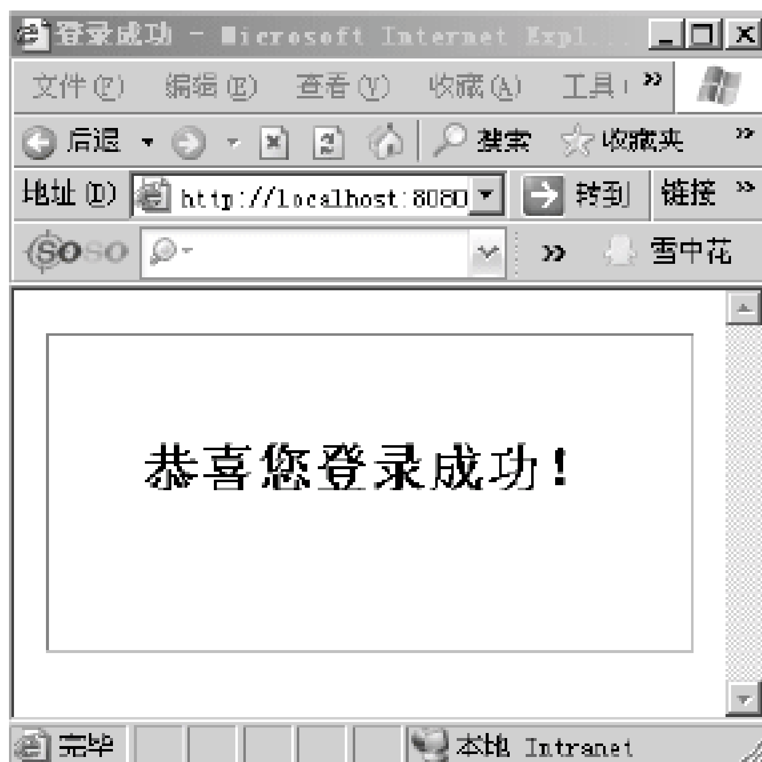


图 18.14　用户登录成功页面

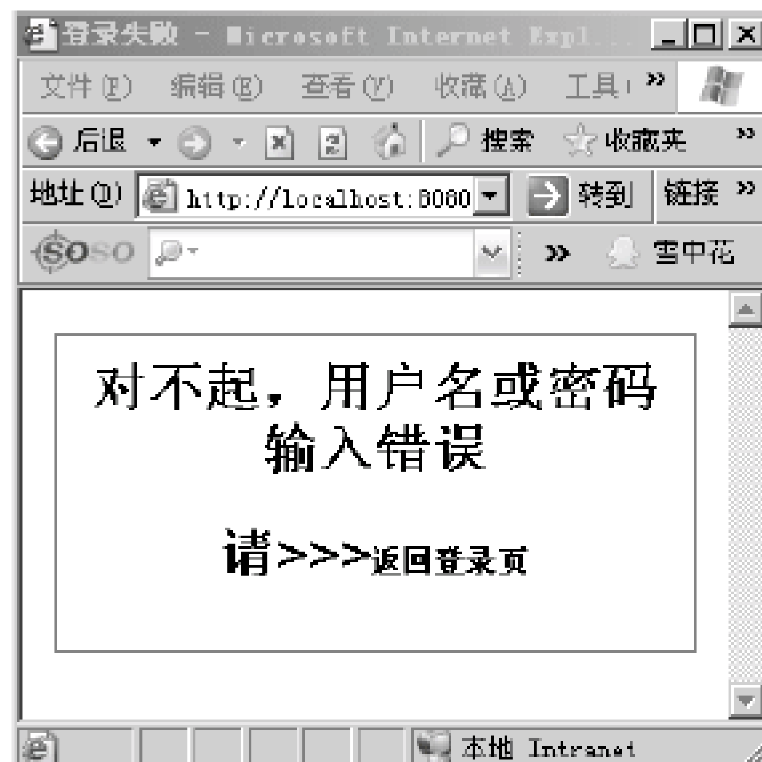


图 18.15　用户登录失败页面

关键技术

在配置<bean>时，可以通过配置 autowire 属性使容器为 JavaBean 自动注入依赖对象。例如，将 autowire 属性值设置为 byName 时，可以按照属性名称的方式查找 JavaBean 依赖的对象并为其注入。以本实例为例，类 Validation 中有一个属性 user，在对类 Validation 进行配置时，指定其 autowire 属性为 byName 后，Spring IoC 容器会在配置文件中查找 id/name 属性为 user 的 bean，然后使用 setter 方法为其注入。

（1）创建 User 对象，定义用户名和密码属性。代码如下：

```
public class User {
        private String username;                                    //用户名
        private String password;                                    //密码
        ……                                                          //省略的 setter 和 getter 方法
}
```

（2）创建控制器 Validation，注入 User 对象并进行登录验证。代码如下：

```
public class Validation extends AbstractController {
private User user;                                                  //注入 User 对象
……                                                                  //省略的 setter 和 getter 方法
 //登录验证：当使用 AbstractController 类作为控制器的父类时
 //只需要改写 handleRequestInternal()方法，实现业务逻辑并返回 ModelAndView 对象
protected ModelAndView handleRequestInternal(HttpServletRequest request,
                HttpServletResponse response) throws Exception {
        String username=arg0.getParameter("username");              //从页面获取用户名的值
        String password=arg0.getParameter("password");              //从页面获取密码的值
        if(username.equals(user.getUsername())&&password.equals(user.getPassword())){
                return new ModelAndView("yes");                     //跳转到登录成功页面
        }else{
                return new ModelAndView("error");                   //跳转到登录失败页面
        }
}
}
```

（3）在 Spring 的配置文件 applicationContext.xml 中为 User 对象的属性赋值，并使用自动装配的方式在控制器 Validation 中注入 User 对象。代码如下：

```
<!--定义控制器转发视图类 -->
<bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix">
                <value>/</value>
        </property>
        <property name="suffix">
                <value>.jsp</value>
        </property>
</bean>
```

```
<!-- 为 User 对象属性赋值 -->
<bean id="user" class="com.user.User">
        <property name="username">
                <value>admin</value>      //设定用户名
        </property>
        <property name="password">
                <value>111</value>        //设定密码
        </property>
</bean>
<!-- 让 bean Validation 自动装配 User 并映射 do -->
<bean name="/login.do"    autowire="byName" class="com.validation.Validation">
        <property name="user">
                <ref local="user"/>
        </property>
</bean>
```

（4）在 web.xml 文件中配置 applicationContext.xml 的自动加载，当项目启动后程序将自动加载配置文件中的信息。代码如下：

```
<!-- 自动加载 Spring 的配置文件 -->
  <servlet>
        <servlet-name>dispatcherServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>/WEB-INF/applicationContext.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
        <servlet-name>dispatcherServlet</servlet-name>
        <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

■ 秘笈心法

心法领悟 475：配置控制器类和配置定义控制器转发视图类。

当使用 AbstractController 作为控制器的父类时，只需要改写 handleRequestInternal()方法，实现业务逻辑，并返回 ModelAndView 对象。在配置定义控制器转发视图类 InternalResourceViewResolver 时，指定 property name 属性为 prefix，表示视图文件的前缀，即目录名（因为把页面放到了目录/WebRoot 之下，所以只需要配置一个“/”；如果把页面放在/WebRoot 目录下一个名为 View 的目录中，这里的 preifx 的值应该为/View/）；指定 property name 属性为 suffix，则表示视图文件的后缀名，即扩展名（例如，如果使用 JSP 文件，则 suffix 的值为.jsp）。

18.2　Spring 的事务管理

实例 476	应用编程式事务管理向用户信息表插入数据 光盘位置：光盘\MR\18\476	高级 实用指数：★★★

■

事务管理对应用程序起着至关重要的作用，应用事务可以保证数据的完整一致性。编程式的事务管理（Programmatic Transaction Management）可以清楚地控制事务边界，也就是让用户自行实现事务的开始时间、撤销操作的时机、结束时间等，可以实现细粒度的事务控制。TransactionTemplate 类是 Spring 提供的事务模板，其编程式事务管理是对 Spring 编程式事务管理最原始方式的封装，使得编码更简单、清晰。本实例将通过 TransactionTemplate 实现向用户信息表中插入数据。运行程序，在如图 18.16 所示页面中输入用户信息，然后单击“添加”按钮，即可在数据库中看到用户信息已被插入，如图 18.17 所示。



图 18.16　填写用户信息

id	name	dept	telephone
1	张*	市场部	138****888
2	李*	销售部	138****777
3	王*	销售部	137****888
4	王**	技术部	138****

图 18.17　在数据库中查看用户信息

关键技术

定义 TransactionTemplate 模板，Bean 中定义了两个属性，一个是 transactionManager 属性，另一个是 propagationBehaviorName 属性。TransactionTemplate 模板通过 transactionManager 属性来处理平台具体的事务管理细节。本例中注册了一个名为 transactionManager 的 Bean 的引用，这个 Bean 是 DataSourceTransactionManager 接口的实现。propagationBehaviorName 属性设置事务传播行为类型，其值为 PROPAGATION_REQUIRED，如果当前没有事务，就新建一个事务；如果已经存在一个事务，则加入到这个事务中。示例代码如下：

```
<bean id="transactionTemplate" class="org.springframework.transaction.support.TransactionTemplate">
        <property name="transactionManager">
                <ref bean="transactionManager"/>
        </property>
        <property name="propagationBehaviorName">
                <value>PROPAGATION_REQUIRED</value>
        </property>
</bean>
```

设计过程

（1）在 Spring 的配置文件中声明事务管理器和 TransactionTemplate。代码如下：

```
<!-- 定义 TransactionTemplate 模板 -->
<bean id="transactionTemplate" class="org.springframework.transaction.support.TransactionTemplate">
        <property name="transactionManager">
                <ref bean="transactionManager"/>
        </property>
        <property name="propagationBehaviorName">
                <value>PROPAGATION_REQUIRED</value>
        </property>
</bean>
<!-- 定义事务管理器 -->
<bean id="transactionManager"
        class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource">
                <ref bean="dataSource" />
        </property>
</bean>
```

（2）创建 TransactionExample 类，定义数据添加的方法。代码如下：

```
public class TransactionExample {
DataSource dataSource;                                                  //注入数据源
PlatformTransactionManager transactionManager;                          //注入事务管理器
TransactionTemplate transactionTemplate;                                //注入 TransactionTemplate 模板
……                                                                      //省略的 setter 和 getter 方法
public void transactionOperation(final String sql){
        transactionTemplate.execute(new TransactionCallback(){
                public Object doInTransaction(TransactionStatus status) {
                        Connection conn = DataSourceUtils.getConnection(dataSource);    //获得数据库连接
                        try {
                                Statement stmt = conn.createStatement();
                                stmt.execute(sql);                              //执行添加方法
                                System.out.println("操作执行成功！ ");
                        } catch (Exception e) {
```

```
                        System.out.println("操作执行失败！ ");
                        System.out.println("原因："+e.getMessage());
                    }
                    return null;
                }
            });
    }
}
```

（3）在 Spring 的配置文件中配置数据源，并为 TransactionExample 注入数据源、事务管理器、TransactionTemplate 模板。代码如下：

```
<!-- 配置数据源 -->
<bean id="dataSource"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName">
                <value>com.mysql.jdbc.Driver</value>
        </property>
        <property name="url">
                <value>jdbc:mysql://localhost:3306/db_database05
                </value>
        </property>
        <property name="username">
                <value>root</value>
        </property>
        <property name="password">
                <value>111</value>
        </property>
</bean>
<!-- 为 TransactionExample 注入数据源 、事务管理器、TransactionTemplate 模板-->
<bean id="transactionExample"
        class="com.transaction.TransactionExample">
        <property name="dataSource">
                <ref bean="dataSource" />
        </property>
        <property name="transactionManager">
                <ref bean="transactionManager" />
        </property>
        <property name="transactionTemplate">
                <ref bean="transactionTemplate"/>
        </property>
</bean>
```

（4）创建一个名为 UserServlet 的 Servlet 类，在 UserServlet 类中定义一个 doM()方法用来获得表单信息、生成插入语句、装载配置文件、执行添加方法。代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        String username = request.getParameter("username");                    //从页面获得用户名的值
        String dept = request.getParameter("dept");                            //从页面获得部门的值
        String telephone = request.getParameter("telephone");                  //从页面获得电话的值
        String sql = "insert into tb_works(name,dept,telephone) values('"+ username + "','" + dept + "','" + telephone + "')";     //生成 SQL 语句
        Resource resource = new ClassPathResource("applicationContext.xml");   //装载配置文件
        BeanFactory factory = new XmlBeanFactory(resource);
        TransactionExample transactionExample = (TransactionExample) factory
                        .getBean("transactionExample");                        //获取 UserDAO
        transactionExample.transactionOperation(sql);                          //执行添加方法

}
```

心法领悟 476：编程式事务管理。

在 Spring 中主要有两种编程式事务管理的实现方法，分别是使用 PlatformTransactionManager 接口的事务管理器和 TransactionTemplate 来实现。虽然二者各有优缺点，但是推荐使用 TransactionTemplate 实现方式，因为它符合 Spring 的模板模式。TransactionTemplate 模板和 Spring 的其他模板一样，封装了资源的打开和关闭等常用的重复代码，在编写程序时只需完成需要的业务代码即可。

实例 477　应用编程式事务管理向学生信息表插入数据

光盘位置：光盘\MR\18\477

高级

实用指数：★★★

实例说明

大多数 Spring 用户选择声明式事务管理器（Declarative Transaction Management），是因为它对应用程序代码影响比较小，也最符合非侵入式轻量级容器的理念。在 Spring 中常用 TransactionProxyFactoryBean 完成声明式事务管理。本实例是通过 TransactionProxyFactoryBean 实现的。从 JSP 页面获取要插入的学生信息（如图 18.18 所示），单击“添加”按钮即可实现向数据库中添加数据。此时在数据库中可以看到数据插入成功，如图 18.19 所示。



图 18.18　填写学生信息

StuId	StuName	StuClass
1	张三	一年五班
2	李四	二年一班

图 18.19　在数据库中查看学生信息

关键技术

在配置文件中定义 TransactionProxyFactoryBean 的步骤如下。

（1）以内部类的形式指定代理的目标对象。

```
<property name="target">
            <bean id="addDAO" class="com.dao.AddDAO">
                  <property name="dataSource">
                        <ref local="dataSource" />
                  </property>
            </bean>
</property>
```

（2）指定事务性方法，通过正则表达式匹配事务性方法，并指定方法的事务属性，也就是说，代理对象中只要是以 add 开头的方法名必须运行在事务中。

```
<property name="proxyTargetClass" value="true" />
      <property name="transactionAttributes">
            <props>
                  <prop key="add*">PROPAGATION_REQUIRED</prop>
            </props>
</property>
```

设计过程

（1）在配置文件中定义数据源 DataSource 和事务管理器（这个事务管理器被注入到 TransactionProxyFactoryBean 中），设置代理对象和事务属性。此处目标对象的定义是以内部类的方式定义的。配置文件中的关键代码如下：

```
<!-- 定义 TransactionProxy -->
<bean id="transactionProxy"
      class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
      <property name="transactionManager">
            <ref local="transactionManager" />
      </property>
      <property name="target">
            <bean id="addDAO" class="com.dao.AddDAO">
                  <property name="dataSource">
```

```
                    <ref local="dataSource" />
               </property>
          </bean>
     </property>
     <property name="proxyTargetClass" value="true" />
     <property name="transactionAttributes">
          <props>
               <prop key="add*">PROPAGATION_REQUIRED</prop>
          </props>
     </property>
</bean>
```

（2）编写 Students 类，用来存储学生的信息。代码如下：

```
public class Students {
private String StuName;                          //学生姓名
private String StuClass;                         //学生班级
.....                                            //省略的 setter 和 getter 方法
}
```

（3）编写操作数据库的 AddDAO 类，通过传递过来的学生信息生成 SQL 语句，执行插入方法。代码如下：

```
public class AddDAO extends JdbcDaoSupport{
public void AddStudent(Students stu){
     String stuname=stu.getStuName();
     String stuclass=stu.getStuClass();
     String sql="insert into tb_student (StuName,StuClass)values('"+stuname+"','"+stuclass+"')";   //生成插入信息的 SQL 语句
     getJdbcTemplate().execute(sql);                                                               //执行插入方法
}
}
```

（4）编写 StuServlet 类，获取页面传递过来的学生信息并存入到 Students 中，然后把信息传递给 AddDAO 类中的 AddStudent()方法。代码如下：

```
public void doM(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
     Resource resource = new ClassPathResource("applicationContext.xml");          //装载配置文件
     BeanFactory factory = new XmlBeanFactory(resource);
     AddDAO dao=(AddDAO)factory.getBean("transactionProxy");                        //获取 UserDAO
     Students stu=new Students();
     stu.setStuName(request.getParameter("StuName"));
     stu.setStuClass(request.getParameter("StuClass"));
     dao.AddStudent(stu);                                                           //执行添加方法
}
```

秘笈心法

心法领悟 477：声明式事务管理。

Spring 的声明式事务不涉及组件依赖关系，而是通过 AOP 实现事务管理。Spring 本身就是一个容器，相对 EJB 容器而言，Spring 显得更为轻便、小巧。在使用 Spring 的声明式事务时无须编写任何代码，便可实现基于容器的事务管理。Spring 提供了一些可供选择的辅助类，这些辅助类简化了传统的数据库操作流程，在一定程度上减少了工作量，提高了编码效率，所以推荐使用声明式事务。在 Spring 中常用 TransactionProxyFactoryBean 完成声明式事务管理。

18.3　Spring 的面向切面编程

实例 478	利用 Spring AOP 使日志输出与方法分离 光盘位置：光盘\MR\18\478	高级 实用指数：★★★

实例说明

对方法进行日志输出是一种很常见的基本功能。传统的做法是把输出语句写在方法体的内部，在调用该方

法时，用输出语句输出信息来记录方法的执行。AOP 可以分离与业务无关的代码，日志输出与方法都做些什么是无关的，其主要目的是记录方法被执行过。本例将利用 Spring AOP 使日志输出与方法分离，使得在调用目标方法之前执行日志输出。程序运行后，在控制台输出的效果如图 18.20 所示。

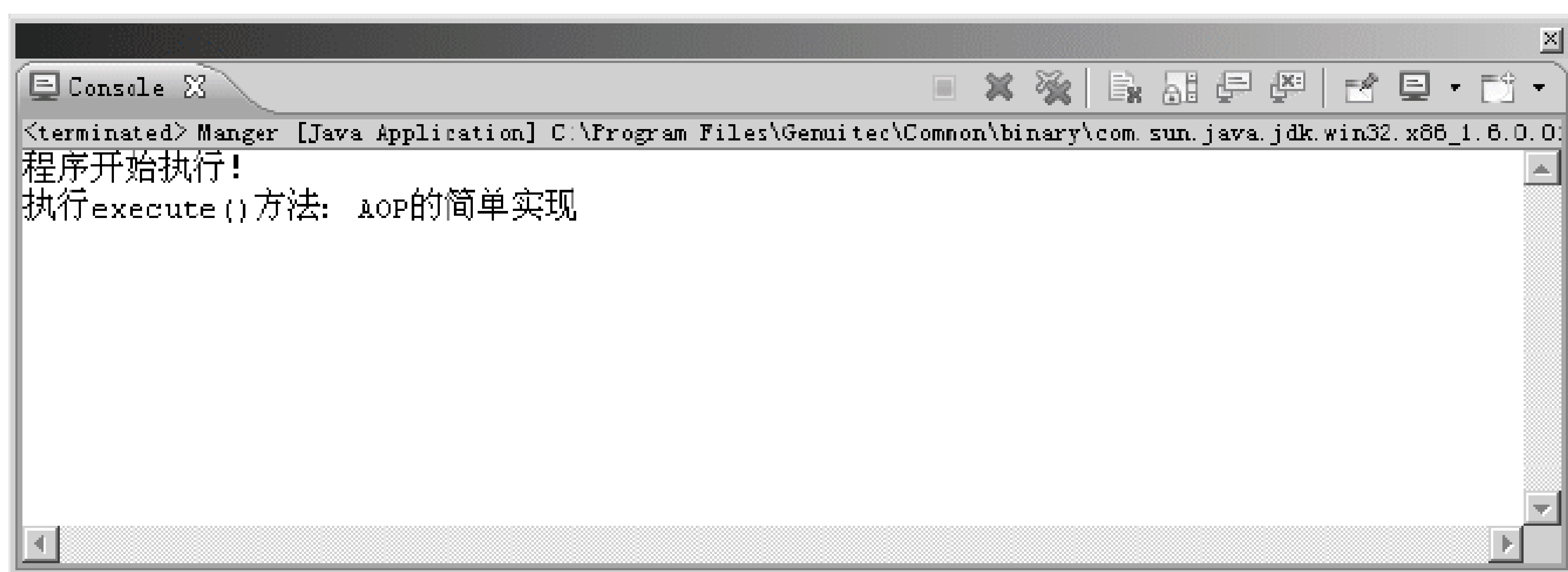


图 18.20　控制台输出的前置通知信息

关键技术

若想使用 AOP 功能，必须创建代理。可以用代码创建代理，代码如下：

```
public static void main(String[] args) {
        Target target = new Target();                    //创建目标对象
      //创建代理
        ProxyFactory di=new ProxyFactory();
        di.addAdvice(new LoggerExecute());
        di.setTarget(target);
        Target proxy=(Target)di.getProxy();
        proxy.execute("AOP 的简单实现");                  //代理执行 execute()方法
    }
```

（1）创建 Target 类作为被代理的目标对象。其中有一个 execute()方法，现在使用 AOP 对 execute()方法做日志输出。执行 execute()方法前，做日志输出。代码如下：

```
public class Target {
    //程序执行的方法
    public void execute(String name){
        System.out.println("执行 execute()方法：" + name);     //输出信息
    }
}
```

（2）创建 LoggerExecute 类，用于通知可以拦截目标对象的 execute()方法，并执行日志输出。代码如下：

```
public class LoggerExecute implements MethodInterceptor {
    public Object invoke(MethodInvocation invocation) throws Throwable {
        before();                                        //执行前置通知
        invocation.proceed();
        return null;
    }
    private void before() {                              //前置通知
        System.out.println("程序开始执行！ ");
    }
}
```

（3）创建 Manger 类，在该类主方法中创建目标对象、创建代理、代理执行 execute()方法。代码如下：

```
public class Manger {

    public static void main(String[] args) {
        Target target = new Target();                //创建目标对象
        ProxyFactory di=new ProxyFactory();          //创建代理
        di.addAdvice(new LoggerExecute());
        di.setTarget(target);
        Target proxy=(Target)di.getProxy();
        proxy.execute("AOP 的简单实现");              //代理执行 execute()方法
    }
}
```

秘笈心法

心法领悟 478：proceed()方法。

invocation 为 MethodInvocation 类型，invocation.proceed()中的 proceed()是执行目标对象的 execute()方法。代码如下：

```
public Object invoke(MethodInvocation invocation) throws Throwable {
        before();                               //执行前置通知
        invocation.proceed();
        return null;
    }
    private void before() {                     //前置通知
        System.out.println("程序开始执行！");
    }
```

实例 479　Spring AOP 实现用户注册

光盘位置：光盘\MR\18\479

高级

实用指数：★★★

实例说明

本实例是以一个用户注册的例子来演示实现 Spring AOP 编程全过程。只需要将用户的注册信息插入到数据库即可，实现的功能比较简单。其主要目的是让读者通过简单的操作理解使用 Spring AOP 的思路。程序运行后，在如图 18.21 所示页面中输入用户注册信息，单击“注册”按钮，在控制台输出的效果如图 18.22 所示。



图 18.21　输入用户注册信息



图 18.22　执行用户注册后控制台输出的信息

关键技术

Before 通知与 After 通知的区别如下。

- Before 通知：顾名思义，它会在目标对象的方法执行之前执行。在具体应用中需要实现 org.springframework. aop.MethodBeforeAdvice 接口，并且要重写默认的 before()方法，该方法会在目标对象所指定的方法之前执行。在 before()方法执行完后，如果没有异常，将会接着执行目标对象的方法。
- After 通知：与 Before 通知非常相似，不过它会在目标对象的方法执行之后执行。在具体应用中需要实现 org.springframework.aop.AfterReturningAdvice 接口，还需要实现 AfterReturningAdvice 接口中的 afterReturning()方法，该方法会在目标对象所指定的方法之后执行。

设计过程

（1）创建代理接口，在该接口中声明了 3 个方法，getConn()为连接数据库方法，execute(sql)是执行 SQL 语句的方法，closeConn()为关闭数据库连接方法。代码如下：

```
public interface UserInterface {
        public abstract void getConn();                         //获取数据库连接的方法
        public abstract void executeInsert(String sql);         //执行添加操作
        public abstract void closeConn();                       //关闭数据库连接
}
```

（2）创建数据库管理的抽象类，该抽象类主要实现接口的 getConn()方法完成数据库的连接。首先要继承 UserInterface 接口，然后实现接口中的 getConn()方法，分别在程序中定义一个私有的成员变量 Con 和 Stmt。当连接创建好后，其他对象可以通过调用 getStmt()方法获得数据库连接。关键代码如下：

```
public abstract class ConnClass implements UserInterface {
    private static Logger logger = Logger.getLogger(AfterAdvice.class.getName());
    private Connection Con = null;
    private Statement Stmt = null;
    ……                                                          //省略的 setter 和 getter 方法
    public void getConn() {                                     //获取数据库连接
        String url = "jdbc:mysql://localhost:3306/db_database18";   //连接数据库的 URL
        try {
            Class.forName("com.mysql.jdbc.Driver");             //数据库驱动
            Con = DriverManager.getConnection(url, "root", "111");  //连接数据库
            logger.info("Connection 已经创建!");
            Stmt = Con.createStatement();                       //创建连接状态
            logger.info("Statement 已经创建!");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

（3）创建 Before 通知，该通知会在 execute()方法执行之前执行，目的是创建数据库连接，为插入数据（执行 execute()）做准备。代码如下：

```
public void before(Method arg0, Object[] arg1, Object arg2)
            throws Throwable {
        logger.info("Before 通知开始……");
        if (arg2 instanceof UserInterface) {
            UserInterface di = (UserInterface) arg2;            // arg2 为目标对象
            di.getConn();                                       //调用 getConn()创建连接
        }
        //以下是将 getConn()创建的连接状态传递给 ExecuteInsert 实现类
        ConnClass ci = (ConnClass) arg2;                        //转换为抽象类对象
        ExecuteInsert bi = (ExecuteInsert) arg2;                //转换为实现类对象
        //将连接状态设置给实现类，目的是让 execute()方法执行前先获得连接
        bi.setState(ci.getStmt());
}
```

（4）创建后置通知，该通知会在 execute()方法执行之后执行，目的是关闭数据库的连接。代码如下：

```
public void afterReturning(Object returnValue, Method method, Object[] args, Object target) throws Throwable {
        logger.info("After 通知开始……");                        //利用 log4j 输出信息
        if (method.getName().equals("executeInsert")){
                if ( target instanceof   UserInterface ){      //后置通知执行后关闭数据库连接
                        UserInterface di=(UserInterface) target;
                        di.closeConn();                         //关闭数据库连接
                }
        }
}
```

（5）创建 commitAction 类，该类是一个 Spring MVC 的控制器类，类似于 Struts 中的 Action 类。在这里可以把它理解为一个 Servlet，其功能主要是获得表单数据，然后插入数据库。代码如下：

```
public ModelAndView execute(HttpServletRequest request,
        HttpServletResponse response) throws SQLException,
                ServletException, IOException {
        request.setCharacterEncoding("gbk");                    //设置编码格式
        String username = request.getParameter("username");     //获取用户名
        String password = request.getParameter("password");     //获取密码
        String tel =request.getParameter("tel");                //获取电话
        String sql="insert into tb_user2 (username,password,tel) values('"+username+"','"+password+"','"+tel+"')";     //执行添加操作的 SQL 语句
        System.out.println("..........................");
        myCheckClass.executeInsert(sql);                        //执行添加操作
        System.out.println("..........................");
        Map map = new HashMap();
```

```
        map.put("msg", "用户注册成功");
        return new ModelAndView("index", map);
}
```

秘笈心法

心法领悟 479：AOP。

在开发过程中，开发人员需要编写两类代码。一类是实现业务逻辑相关功能的代码，另一类是与业务逻辑关系不大的代码，如日志、异常处理等。一般情况下，这两类代码会被写在一起。例如，在编写对数据库操作的 DAO 模型时，所有的 DAO 类都需要获取数据库连接、关闭数据库连接。这样的程序十分不利于维护，而 AOP 就是把与业务逻辑关系不大的代码从程序中分离出来，降低代码的耦合性，提高代码重用率。

18.4　Spring 的持久化

实例 480	在 Spring 中利用 DAO 模式添加数据 光盘位置：光盘\MR\18\480	高级 实用指数：★★★

实例说明

DAO 代表数据访问对象（Data Access Object），其主要目的是将持久性相关的问题与一般的业务规则和工作流隔离开来，为定义业务层可以访问的持久性操作引入了一个接口并且隐藏了实现的具体细节，该接口的功能将依赖于采用的持久性技术而改变，但是 DAO 接口可以基本上保持不变。为了更深入地理解 Spring 的 DAO 模式，下面介绍如何利用 Spring 的 DAO 模式向商品信息表中添加数据。运行程序后，在如图 18.23 所示页面中输入商品信息，单击“添加到数据库”按钮，商品信息就会被保存到数据库中，如图 18.24 所示。



图 18.23　输入商品信息

id	name	price	type
1	火腿肠	2	食品

图 18.24　数据库中的商品信息表

关键技术

创建 DAO 接口 GoodsDao 类，定义一个名为 addGoods()的插入方法，再编写 GoodsDaoImpl 类实现接口中的 addGoods()方法。这样就实现了通过接口提供对外服务，程序的其他模块将通过这些接口来访问数据库。这样做有很多好处：首先，由于服务对象不再和特定的接口实现绑定在一起，使其易于测试，因为它提供的是一种服务，在不需要连接数据库的条件下就可以进行单元测试，极大地提高了开发效率；其次，通过使用与持久化技术无关的方法访问数据库，在应用程序的设计和使用上都有很大的灵活性，对于整个系统无论是在性能上还是应用上也是一个巨大的飞跃。

设计过程

（1）创建名为 GoodsInfo 的 JavaBean 类，用于封装商品信息。代码如下：

```
public class GoodsInfo {
 private int id;                //商品编号
 private String name;           //商品名称
```

```
private float price;                    //商品价格
private String type;                    //商品类别
……                                      //省略的 setter 和 getter 方法
}
```

（2）创建操作商品信息的接口 GoodsDao，并定义添加商品信息的 addGoods()方法，参数类型为 GoodsInfo 实体对象。代码如下：

```
public interface GoodsDao {
        public void addGoods(GoodsInfo goods);
}
```

（3）编写实现此 DAO 接口的 GoodsDaoImpl 类。首先定义一个用于操作数据库的数据源对象 DataSource，通过它创建一个数据库连接对象建立与数据库的连接，GoodsDaoImpl 类实现了接口的 addGoods()方法，在该类中实现访问数据库。代码如下：

```
public class GoodsDaoImpl implements GoodsDao {
private DataSource dataSource;
……                                                                          //省略的 setter 和 getter 方法
public void addGoods(GoodsInfo goods) {
        Connection conn=null;
        PreparedStatement stmt=null;
        try{
                conn = dataSource.getConnection();                          //获取数据库连接
                String sql = "insert into tb_commodity(name,price,type) values(?,?,?);";   //插入商品信息的 SQL 语句
                stmt = conn.prepareStatement(sql);                          //创建预编译对象
                stmt.setString(1, goods.getName());                         //为商品名称赋值
                stmt.setFloat(2, goods.getPrice());                         //为商品价格赋值
                stmt.setString(3, goods.getType());                         //为商品类别赋值
                stmt.executeUpdate();                                       //编译执行，更新数据库
        }catch(Exception ex){
                ex.printStackTrace();
        }
        ……                                                                  //省略了其他代码
    }
}
```

（4）编写 Spring 的配置文件 applicationContext.xml。在该配置文件中，首先定义一个 JavaBean 名称为 DataSource 的数据源，它是 Spring 中的 DriverManagerDataSource 类的实例；然后配置前面编写完的 GoodsDAOImpl 类，并且注入其 DataSource 属性值。代码如下：

```
<!-- 配置数据源 -->
<bean id="dataSource" class=" org.springframework.jdbc.datasource.DriverManagerDataSource ">
        <property name="driverClassName">
                <value>com.mysql.jdbc.Driver</value>
        </property>
        <property name="url" >
                <value>jdbc:mysql:// localhost:3306/db_database18</value>
        </property>
        <property name="username">
                <value>root</value>
        </property>
        <property name="password">
                <value>111</value>
        </property>
</bean>
<!-- 为 GoodsDAO 注入数据源 -->
<bean id="goodsDao" class="com.dao.impl.GoodsDaoImpl">
        <property name="dataSource">
                <ref local="dataSource"/>
        </property>
</bean>
```

（5）创建添加商品信息的表单页 index.jsp，设置表单提交到 save.jsp 处理页。具体代码参见配书光盘。

（6）创建 save.jsp 页。代码如下：

```
<%
request.setCharacterEncoding("GBK");
String name = request.getParameter("name");                 //获取商品名称
String price = request.getParameter("price");               //获取商品价格
```

```
String type = request.getParameter("type");                              //获取商品类别

GoodsInfo goods = new GoodsInfo();                                       //创建商品的 JavaBean
goods.setName(name);                                                     //添加商品名称
goods.setPrice(Float.parseFloat(price));                                 //添加商品价格
goods.setType(type);                                                     //添加商品类别

Resource resource = new ClassPathResource("applicationContext.xml");     //装载配置文件
BeanFactory factory = new XmlBeanFactory(resource);
GoodsDaoImpl dao = (GoodsDaoImpl)factory.getBean("goodsDao");            //获取 Bean 的实例
dao.addGoods(goods);                                                     //调用方法添加商品信息
out.println("<script type='text/javascript'> alert('添加成功！');window.location.href='index.jsp'</script>");
%>
```

秘笈心法

心法领悟 480：PreparedStatement 预编译。

预编译的 SQL 语句是通过“?”来传递值的，这样的 SQL 语句会比普通的 SQL 语句多出几行代码，看似麻烦了很多，其实这样的代码无论从可读性还是可维护性上来说，效率都会提高很多。尤其重要的是，它还大大提高了语句的安全性（防止 SQL 注入）。

实例 481 利用 JdbcTemplate 向员工信息表添加数据

光盘位置：光盘\MR\18\481

高级

实用指数：★★★

实例说明

JdbcTemplate 类是 Spring 的核心类之一。该类在内部已经处理完了数据库资源的建立和释放，并可以避免一些常见的错误，例如，关闭连接、抛出异常等。因此，使用 JdbcTemplate 类简化了编写 JDBC 时所使用的基础代码。下面介绍如何利用 JdbcTemplate 向员工信息表中添加数据。运行程序，在如图 18.25 所示页面中添加员工信息，单击“添加到数据库”按钮，员工信息将被保存到数据库中，如图 18.26 所示。

图 18.25　输入员工信息

id	name	sex	age	dept	duty	telephone
1	王*	男	30	销售部	经理	138*****888

图 18.26　数据库中的员工信息表

关键技术

利用 JdbcTemplate 类进行数据写入主要是通过 update()方法（它实现了很多方法的重载特征）来实现。在本实例中使用了 JdbcTemplate 类写入数据的常用方法 update(String)，代码如下：

```
public void addEmp(Employee emp){
        jdbcTemplate.update(sql);                    //sql 为执行添加的 SQL 语句
}
```

设计过程

（1）创建封装员工信息的 JavaBean 类 Employee。代码如下：

```
public class Employee {
```

```
private int id;                //编号
private String name;           //姓名
private String sex;            //性别
private int age;               //年龄
private String dept;           //部门
private String duty;           //职务
private String telephone;      //联系电话
……                             //省略的 getter 和 setter 方法
}
```

（2）创建操作数据库的 Dao 类 EmpDao，声明一个 JdbcTemplate 类型属性，并添加 getter 和 setter 方法，用于将 JdbcTemplate 注入到 Empdao，然后编写一个向员工信息表添加数据的方法。代码如下：

```
public class EmpDao {
private JdbcTemplate jdbcTemplate;                    //注入 JdbcTemplate
public JdbcTemplate getJdbcTemplate() {
        return jdbcTemplate;
}
public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
}
public void addEmp(Employee emp){                     //向员工信息表添加数据的方法
        String name=emp.getName();
        String sex = emp.getSex();
        int age = emp.getAge();
        String dept = emp.getDept();
        String duty = emp.getDuty();
        String telephone = emp.getTelephone();
        String sql = "insert into tb_employee(name,sex,age,dept,duty,telephone)
                values('"+name+"','"+sex+"',"+age+",'"+dept+"','"+duty+"','"+telephone+"')";
        jdbcTemplate.update(sql);                     //调用 jdbcTemplate 的 update()方法向员工信息表插入数据
}
}
```

（3）在 Spring 的 applicationContext.xml 文件中，首先配置 DataSource 数据源；然后配置 JdbcTemplate，将数据源对象注入 JdbcTemplate 中；最后配置 EmpDao，并将 JdbcTemplate 注入 EmpDao。代码如下：

```
<!-- 配置数据源 -->
<bean id="dataSource"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName">
                <value>com.mysql.jdbc.Driver</value>
        </property>
        <property name="url">
                <value>jdbc:mysql://localhost:3306/db_database18
                </value>
        </property>
        <property name="username">
                <value>root</value>
        </property>
        <property name="password">
                <value>111</value>
        </property>
</bean>
<!-- 配置 jdbcTemplate -->
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource">
                <ref local="dataSource"/>
        </property>
</bean>
<!-- 配置 Dao -->
<bean id="empDao" class="com.dao.EmpDao">
        <property name="jdbcTemplate">
                <ref local="jdbcTemplate"/>
        </property>
</bean>
```

（4）创建填写员工信息的表单页 index.jsp，具体代码参见配书光盘。

（5）创建处理表单数据的 save.jsp 页。代码如下：

```
<%
```

```
String name = request.getParameter("name");                                  //获取表单数据
String sex = request.getParameter("sex");
String age = request.getParameter("age");
String dept = request.getParameter("dept");
String duty = request.getParameter("duty");
String tel = request.getParameter("telephone");
Employee emp = new Employee();                                               //创建封装员工信息的 JavaBean
emp.setName(name);
emp.setSex(sex);
emp.setAge(Integer.parseInt(age));
emp.setDept(dept);
emp.setDuty(duty);
emp.setTelephone(tel);
Resource resource = new ClassPathResource("applicationContext.xml");         //装载配置文件
BeanFactory factory = new XmlBeanFactory(resource);
EmpDao dao = (EmpDao)factory.getBean("empDao");                              //获取 Dao 对象
dao.addEmp(emp);                                                             //调用方法，添加员工信息
%>
```

秘笈心法

心法领悟 481：JdbcTemplate 类。

JdbcTemplate 类可以直接通过数据源的引用实例化，然后在服务中使用，也可以通过依赖注入的方式在 ApplicationContext 中产生并作为 JavaBean 的引用给服务使用。JdbcTemplate 类运行了核心的 JDBC 工作流程，例如，应用程序要创建和执行 Statement 对象，只需在代码中提供 SQL 语句。此外，这个类可以执行 SQL 中的查询、更新或者调用存储过程等操作，同时生成结果集的迭代数据，还可以捕捉 JDBC 的异常并将其转换成 org.springframework.dao 包中定义的通用的能够提供更多信息的异常体系。

实例 482 利用 JdbcTemplate 查询员工信息表

高级

光盘位置：光盘\MR\18\482

实用指数：★★★

实例说明

下面介绍如何利用 JdbcTemplate 查询员工信息表，运行结果如图 18.27 所示。

员 工 信 息 查 询

当前位置>>>所有员工信息

编号	姓名	性别	年龄	部门	职位	电话
1	孙*	男	30	销售部	经理	138*****888
2	张*	女	26	客服部	经理	137*****888
3	李*	男	23	销售部	员工	138****8888

图 18.27　查询员工信息

关键技术

JdbcTemplate 类的 queryForList()方法是 Spring 中封装好的查询方法，返回的是一个 List 集合，集合中的元素为 Map 对象，而 Map 中的 Key 与数据库表的字段相对应。

```
List list=jdbcTemplate.queryForList(sql);
```

设计过程

（1）创建封装员工信息的 JavaBean 类 Employee。关键代码如下：

```
public class Employee {
 private int id;          //编号
 private String name;     //姓名
```

```
private String sex;          //性别
private int age;             //年龄
private String dept;         //部门
private String duty;         //职务
private String telephone;    //联系电话
……                           //省略的 getter 和 setter 方法
}
```

（2）创建操作数据库的 Dao 类 EmpDao，声明一个 JdbcTemplate 类型属性，并添加 getter 和 setter 方法，用于将 JdbcTemplate 注入到 Empdao；然后编写一个查询员工信息表的方法，返回类型是 Iterator。关键代码如下：

```
public class EmpDao {
private JdbcTemplate jdbcTemplate;                                              //注入 JdbcTemplate
public JdbcTemplate getJdbcTemplate() {
        return jdbcTemplate;
}
public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
}
public Iterator selectEmp() {
        String sql = "select id,name,sex,age,dept,duty,telephone from tb_employee";   //生成 SQL 查询语句
        List list=jdbcTemplate.queryForList(sql);                               //执行查询方法赋值给 List
        Iterator it=   list.iterator();                                         //通过 List 集合生成 Iterator
        return it;
}
}
```

（3）在 Spring 的 applicationContext.xml 文件中，首先配置 DataSource 数据源；然后配置 JdbcTemplate，将数据源对象注入 JdbcTemplate 中；最后配置 EmpDao，并将 JdbcTemplate 注入 EmpDao。关键代码如下：

```
<!-- 配置 jdbcTemplate -->
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource">
                <ref local="dataSource"/>
        </property>
</bean>
<!-- 配置 Dao -->
<bean id="empDao" class="com.dao.EmpDao">
        <property name="jdbcTemplate">
                <ref local="jdbcTemplate"/>
        </property>
</bean>
```

（4）创建显示员工信息的表单页 index.jsp，具体代码参见配书光盘。

秘笈心法

心法领悟 482：获取 Dao 对象。

当调用查询方法时，操作数据库的 Dao 类 EmpDao 是从 XML 文件中获取的，而不是直接 new 出来，写代码时要注意。

```
Resource resource = new ClassPathResource("applicationContext.xml");        //装载配置文件
BeanFactory factory = new XmlBeanFactory(resource);
EmpDao dao = (EmpDao)factory.getBean("empDao");                            //获取 Dao 对象
Iterator it=dao.selectEmp();                                               //调用方法，查询员工信息
```

实例 483	利用 JdbcTemplate 更新指定员工信息 光盘位置：光盘\MR\18\483	高级 实用指数：★★★

本例将介绍如何利用 JdbcTemplate 更新指定员工信息。运行程序，在如图 18.28 所示页面中选择想要更新的员工，单击“修改”超链接，跳转到如图 18.29 所示修改页面，修改员工信息后单击“修改员工信息”按钮，即可执行员工信息修改操作。



图 18.28　所有员工信息页面

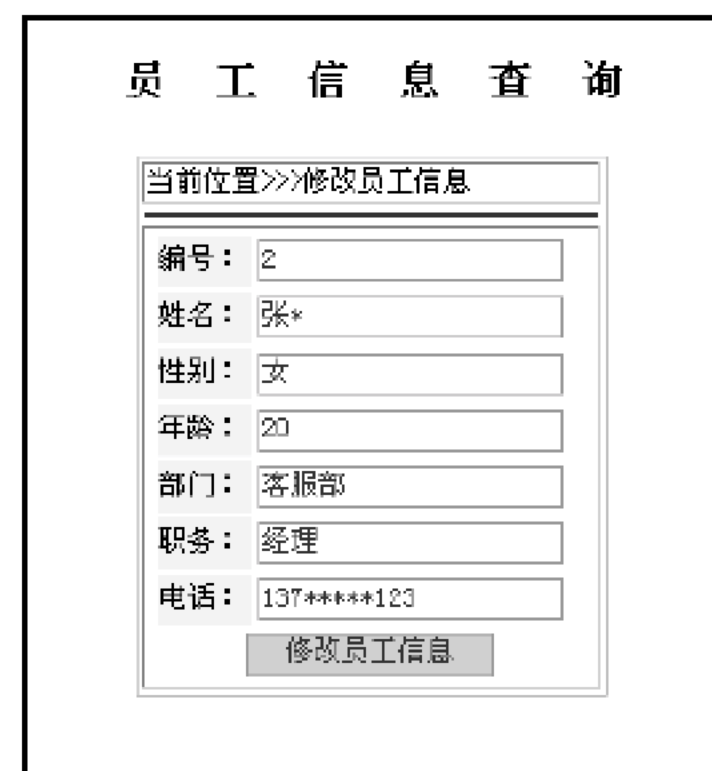


图 18.29　修改员工信息页面

■ 关键技术

JdbcTemplate 类的 update()方法是 Spring 封装好的修改方法，调用时只需要传递 sql 即可执行修改操作。代码如下：

```
jdbcTemplate.update(sql);
```

■ 设计过程

（1）创建封装员工信息的 JavaBean 类 Employee，关键代码如下：

```
public class Employee {
 private int id;              //编号
 private String name;         //姓名
 private String sex;          //性别
 private int age;             //年龄
 private String dept;         //部门
 private String duty;         //职务
 private String telephone;    //联系电话
 ……                           //省略的 getter 和 setter 方法
}
```

（2）创建操作数据库的 Dao 类 EmpDao，声明一个 JdbcTemplate 类型属性，并添加 getter 和 setter 方法，用于将 JdbcTemplate 注入到 Empdao；然后编写一个查询所有员工信息的方法，返回类型是 Iterator；再编写一个获取单个员工信息的方法；最后编写一个执行修改信息的方法。关键代码如下：

```
public class EmpDao {
 private JdbcTemplate jdbcTemplate;

 public JdbcTemplate getJdbcTemplate() {
        return jdbcTemplate;
 }

 public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
 }
 public Iterator selectEmp() {                                          //查询所有员工信息
        String sql = "select id,name,sex,age,dept,duty,telephone from tb_employee";  //生成 SQL 查询语句
        List list=jdbcTemplate.queryForList(sql);                      //执行查询方法赋值给 List
        Iterator it= list.iterator();                                  //通过 List 集合生成 Iterator
        return it;
 }
 public Iterator getOneEmp(int id) {                                    //查询单个员工信息
        String sql = "select id,name,sex,age,dept,duty,telephone from tb_employee where id="+id+"";  //生成 SQL 查询语句
        List list=jdbcTemplate.queryForList(sql);                      //执行查询方法赋值给 List
        Iterator it= list.iterator();                                  //通过 List 集合生成 Iterator
        return it;
 }
 public void update(Employee emp){                                    //修改员工信息
        int id=emp.getId();                                           //获取 id
```

```
        String name=emp.getName();                                              //获取姓名
        String sex = emp.getSex();                                              //获取性别
        int age = emp.getAge();                                                 //获取年龄
        String dept = emp.getDept();                                            //获取部门
        String duty = emp.getDuty();                                            //获取职位
        String telephone = emp.getTelephone();                                  //获取电话
        String sql="update tb_employee set name='"+name+"',sex='"+sex+"',age="+age+",dept='"+dept+"',duty='"+duty+"',telephone='"+telephone+"'
where id="+id+"";                                                               //生成修改语句
        jdbcTemplate.update(sql);                                               //执行修改方法
}
}
```

（3）在 Spring 的 applicationContext.xml 文件中，首先配置 DataSource 数据源；然后配置 jdbcTemplate，将数据源对象注入 jdbcTemplate 中；最后配置 EmpDao，并将 jdbcTemplate 注入 EmpDao。关键代码如下：

```
<!-- 配置 jdbcTemplate -->
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource">
                <ref local="dataSource"/>
        </property>
</bean>
<!-- 配置 Dao -->
<bean id="empDao" class="com.dao.EmpDao">
        <property name="jdbcTemplate">
                <ref local="jdbcTemplate"/>
        </property>
</bean>
```

（4）创建显示员工信息的表单页 index.jsp，具体代码参见配书光盘。

（5）创建修改员工信息的表单页 getone.jsp，具体代码参见配书光盘。

（6）创建处理表单数据的 update.jsp 页。代码如下：

```
<%
int id =Integer.parseInt(request.getParameter("id")) ;                          //获取修改的信息
String name = request.getParameter("name");
String sex = request.getParameter("sex");
String age = request.getParameter("age");
String dept = request.getParameter("dept");
String duty = request.getParameter("duty");
String tel = request.getParameter("telephone");
Employee emp = new Employee();
emp.setId(id);                                                                  //为 emp 对象赋值
emp.setName(name);
emp.setSex(sex);
emp.setAge(Integer.parseInt(age));
emp.setDept(dept);
emp.setDuty(duty);
emp.setTelephone(tel);
Resource resource = new ClassPathResource("applicationContext.xml");            //装载配置文件
BeanFactory factory = new XmlBeanFactory(resource);
EmpDao dao = (EmpDao)factory.getBean("empDao");                                 //获取 Dao 对象
dao.update(emp);                                                                //调用方法，修改员工信息
out.println("<script type='text/javascript'> alert('修改成功！');window.location.href='index.jsp'</script>");
%>
```

心法领悟 483：Get 和 Post。

提交表单的方式有两种，即 Get 和 Post。其中，Get 是默认的提交方式，如果不在 method()方法中声明，表单就会以 Get 方式提交到 action 属性所指的 URL 中，提交的数据会显示到浏览器地址栏内，并且 Get 限制 Form 表单的数据集为 ASCII 字符集，而 ASCII 是不支持中文的；而 Post 方式提交表单后的地址栏不变，并且它支持 ISO10646 字符集，所以当提交表单内容带有中文时应选择 Post 方式。

```
<form action="update.jsp" method="post">
```



实例 484 使用 JdbcTemplate 调用存储过程查询商品

光盘位置：光盘\MR\18\484

高级

实用指数：★★★

实例说明

本实例将演示如何利用 Spring 的 JdbcTemplate 来调用数据库的存储过程查询商品信息。运行程序，在页面中显示出查询到的指定类别的所有商品信息列表，如图 18.30 所示。

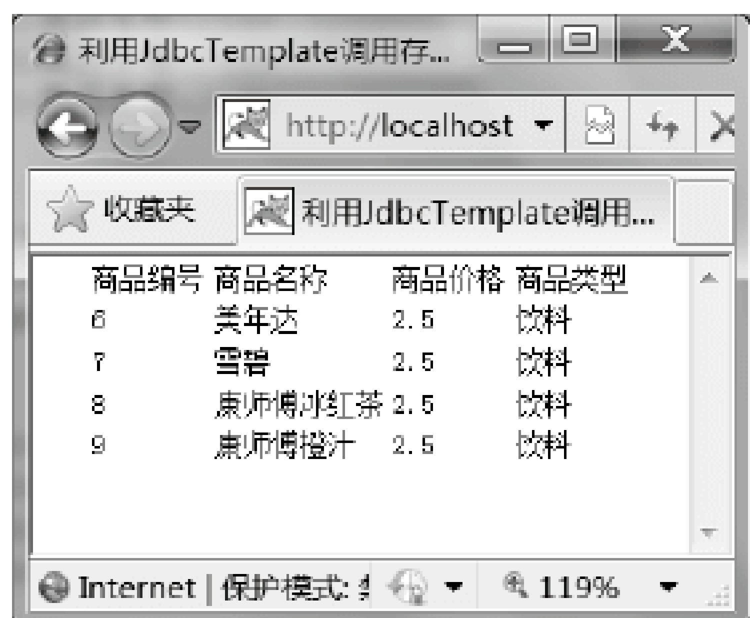


图 18.30　调用存储过程查询商品信息

关键技术

在实现调用存储过程时，主要应用的是 JdbcTemplate 类的 execute()方法。该方法包含多个重载方法，本实例中主要应用的是 execute(String callString, CallableStatementCallback action)方法。其语法如下：

```
public Object execute(String callString, CallableStatementCallback action)
```

参数说明

❶ callString：指定调用存储过程的 SQL 语句。

❷ action：CallableStatementCallback 接口类型，它是 Spring 提供的用于处理存储过程的回调接口。此处需要实现这个接口的 doInCallableStatement()方法来执行调用存储过程。

（1）在 MySQL 数据库中，创建根据商品类别查询商品信息的存储过程。代码如下：

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `selectGoodsInfo` $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `selectGoodsInfo`(IN goodsType varchar(1000))
BEGIN
      set @flag = goodsType;
      select * from tb_goodsinfo where species=@flag;

END $$

DELIMITER ;
```

（2）创建 GoodsInfo 类，用于封装商品信息。代码如下：

```
public class GoodsInfo {
 private int id;
 private String goodsName;
 private float price;
 private String type;
 private String remark;
 ……
}
```

（3）创建 GoodsDao 类，在该类中添加 JdbcTemplate 的私有属性，并设置 getter 和 setter 方法，因为此处需要通过 Spring 的配置将 JdbcTemplate 的对象注入 GoodsDao 中，然后编写 selectGoodsInfo()方法，在该方法中实现调用存储过程查询商品信息。代码如下：

```
public Object selectGoodsInfo(final String type){
        List<SqlParameter> parameters = new ArrayList<SqlParameter>();
        parameters.add(new SqlParameter(Types.VARCHAR));

        Object result = jdbcTemplate.execute("call selectGoodsInfo(?)",new CallableStatementCallback(){//回调接口
              public Object doInCallableStatement(CallableStatement cs)
                              throws SQLException, DataAccessException {
                    cs.setString(1, type);                          //设置存储过程的参数
```

```
                ResultSet rs = cs.executeQuery();              //执行存储过程//返回结果集
                List list = new ArrayList();                   //创建 List 集合
                while(rs.next()){                              //循环结果集，将结果集数据封装到对象中
                    GoodsInfo goods = new GoodsInfo();         //创建商品对象
                    goods.setId(rs.getInt(1));
                    goods.setGoodsName(rs.getString(2));
                    goods.setPrice(rs.getFloat(3));
                    goods.setType(rs.getString(4));
                    list.add(goods);                           //将数据对象保存到集合中
                }
                return list;
            }});
        return result;                                         //返回存储数据的对象，该 Object 对象就是一个 List 集合
}
```

（4）在 Spring 的配置文件中配置数据源，并配置将 JdbcTemplate 的对象注入到 GoodsDao 中。具体代码参见配书光盘。

（5）创建 index.jsp 表单页，根据商品类别查询商品信息。代码如下：

```
<form action="select.jsp" method="post">
    商品类别：<input type="text" name="type" />
    <input type="submit" value="查询" />
</form>
```

（6）创建 select.jsp 页，应用 BeanFactory 获取 Bean 对象，然后调用存储过程将查询结果输出到页面。代码如下：

```
<%
String type = request.getParameter("type");
Resource resource = new ClassPathResource("applicationContext.xml");    //装载配置文件
BeanFactory factory = new XmlBeanFactory(resource);
GoodsDao dao = (GoodsDao)factory.getBean("goodsDao");                   //获取 Dao 对象
Object result = dao.selectGoodsInfo(type);
%>
<center>
    <table>
        <tr>
            <td>商品编号</td>
            <td>商品名称</td>
            <td>商品价格</td>
            <td>商品类型</td>
        </tr>
<%
List list =(ArrayList)result;
for(int i=0;i<list.size();i++){
    GoodsInfo goods = (GoodsInfo)list.get(i);
        %>
        <tr>
            <td><%=goods.getId() %></td>
            <td><%=goods.getGoodsName() %></td>
            <td><%=goods.getPrice() %></td>
            <td><%=goods.getType() %></td>
        </tr>
        <%
    }
%>
    </table>
</center>
```

秘笈心法

心法领悟 484：调用存储过程的其他方法。

在 JdbcTemplate 类中，还有几个模板方法可以实现调用存储过程，如 execute(CallableStatementCreator arg0 ,CallableStatementCallback arg1)方法和 call(CallableStatementCreator,List<SqlParameter> sqlParam)方法，这些方法的详细说明参见 Spring 提供的 API，在此不详细说明。在实际应用中，可以根据不同的情况选择不同的实现方法。



实例 485 使用 SimpleJdbcTemplate 添加图书信息

光盘位置：光盘\MR\18\485

高级

实用指数：★★★

实例说明

本例中将介绍如何利用 SimpleJdbcTemplate 添加图书信息。运行程序，在如图 18.31 所示的页面中输入图书信息，单击“添加”按钮，即可完成添加操作。

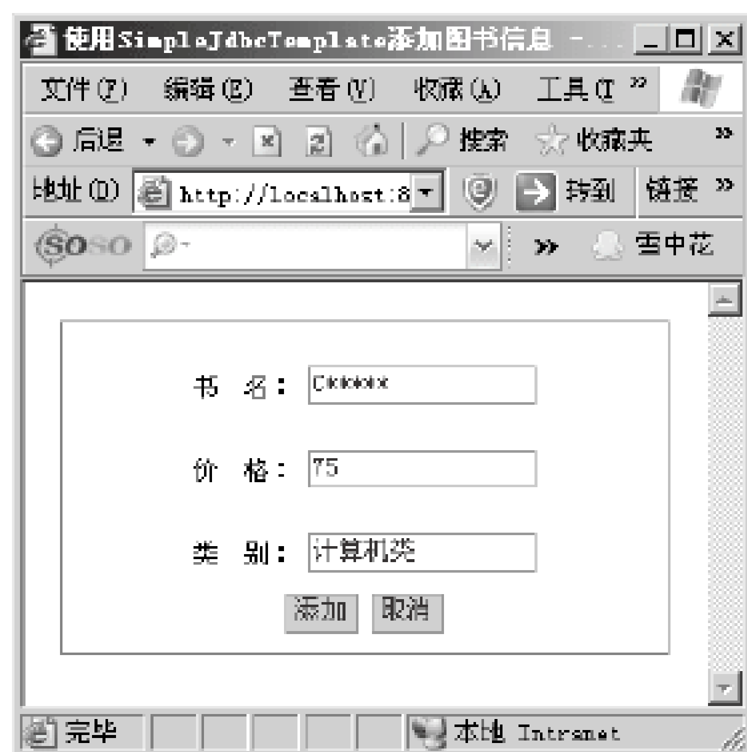


图 18.31　添加图书信息

关键技术

SimpleJdbcTemplate 类的 update()方法是 Spring 封装好的添加方法。调用该方法时，需要传递 3 个参数，分别为 SQL 语句、SqlParameterSource 和 KeyHolder。代码如下：

```
simpleJdbcTemplate.getNamedParameterJdbcOperations().update(String sql，SqlParameterSource param,Keyholder keyholder);
```

（1）创建封装图书信息的 JavaBean 类 Books，关键代码如下：

```
public class Books {
    private int id;              //图书编号
    private String name;         //图书名称
    private float total;         //图书价格
    private String species;      //图书种类
    ……                          //省略的 getter 和 setter 方法
}
```

（2）创建操作数据库的 Dao 类 BookDao，声明一个 simpleJdbcTemplate 类型属性，并添加 getter 和 setter 方法，用于将 simpleJdbcTemplate 注入到 BookDao；然后编写一个添加图书信息的方法。关键代码如下：

```
public class BookDao {
    private SimpleJdbcTemplate simpleJdbcTemplate;                                    //注入 simpleJdbcTemplate
    ……                                                                                //省略的 getter 和 setter 方法
    public void addbook(Books book){
        String name=book.getName();
        float total=book.getTotal();
        String species=book.getSpecies();
        String sql = "insert into tb_books(name,total,species) values('"+name+"','"+total+"','"+species+"')";    //生成添加的 SQL 语句
        SqlParameterSource param=new BeanPropertySqlParameterSource(book);
        KeyHolder keyholder=new GeneratedKeyHolder();
        simpleJdbcTemplate.getNamedParameterJdbcOperations().update(sql, param, keyholder);      //添加方法
    }
}
```

（3）在 Spring 的 applicationContext.xml 文件中，首先配置 DataSource 数据源；然后配置 simpleJdbcTemplate，将数据源对象注入 simpleJdbcTemplate 中；最后配置 BookDao，并将 simpleJdbcTemplate 注入 BookDao。关键代码如下：

```
<!-- 配置 simplejdbcTemplate -->
<bean id="simpleJdbcTemplate" class="org.springframework.jdbc.core.simple.SimpleJdbcTemplate">
```

```
        <constructor-arg><ref bean="dataSource" /></constructor-arg>
</bean>
<!-- 配置 Dao -->
<bean id="bookDao" class="com.dao.BookDao">
        <property name="simpleJdbcTemplate">
                <ref local="simpleJdbcTemplate"/>
        </property>
</bean>
```

（4）创建输入图书信息的表单页 index.jsp，具体代码参见配书光盘。

（5）创建添加图书信息的表单页 save.jsp，关键代码如下：

```
<%
request.setCharacterEncoding("GBK");                              //获取图书的信息
String name = request.getParameter("name");                       //书名
String total = request.getParameter("total");                     //价格
String species = request.getParameter("species");                 //类别
Books book = new Books();
book.setName(name);
book.setTotal((float)Integer.parseInt(total));
book.setSpecies(species);
Resource resource = new ClassPathResource("applicationContext.xml");  //装载配置文件
BeanFactory factory = new XmlBeanFactory(resource);
BookDao dao = (BookDao)factory.getBean("bookDao");                //获取 Dao 对象
dao.addbook(book);                                                //添加图书信息
out.println("<script type='text/javascript'> alert('添加成功！');window.location.href='index.jsp'</script>");
%>
```

秘笈心法

心法领悟 485：SimpleJdbcTemplate。

SimpleJdbcTemplate 只是提供了 JdbcTemplate 所提供功能的子类，当需要使用 JdbcTemplate 的方法，而这些方法没有在 SimpleJdbcTemplate 中定义时，则需要调用 getJdbcOperations()方法获取相应的方法调用。

实例 486	使用 SimpleJdbcTemplate 查询指定图书信息 光盘位置：光盘\MR\18\486	高级 实用指数：★★★★

实例说明

本例将介绍如何利用 SimpleJdbcTemplate 查询指定图书信息。运行程序，在如图 18.32 所示页面中选择想要查看的图书，然后单击该记录后面的“查看详情”超链接，即可查看该书详细信息，如图 18.33 所示。

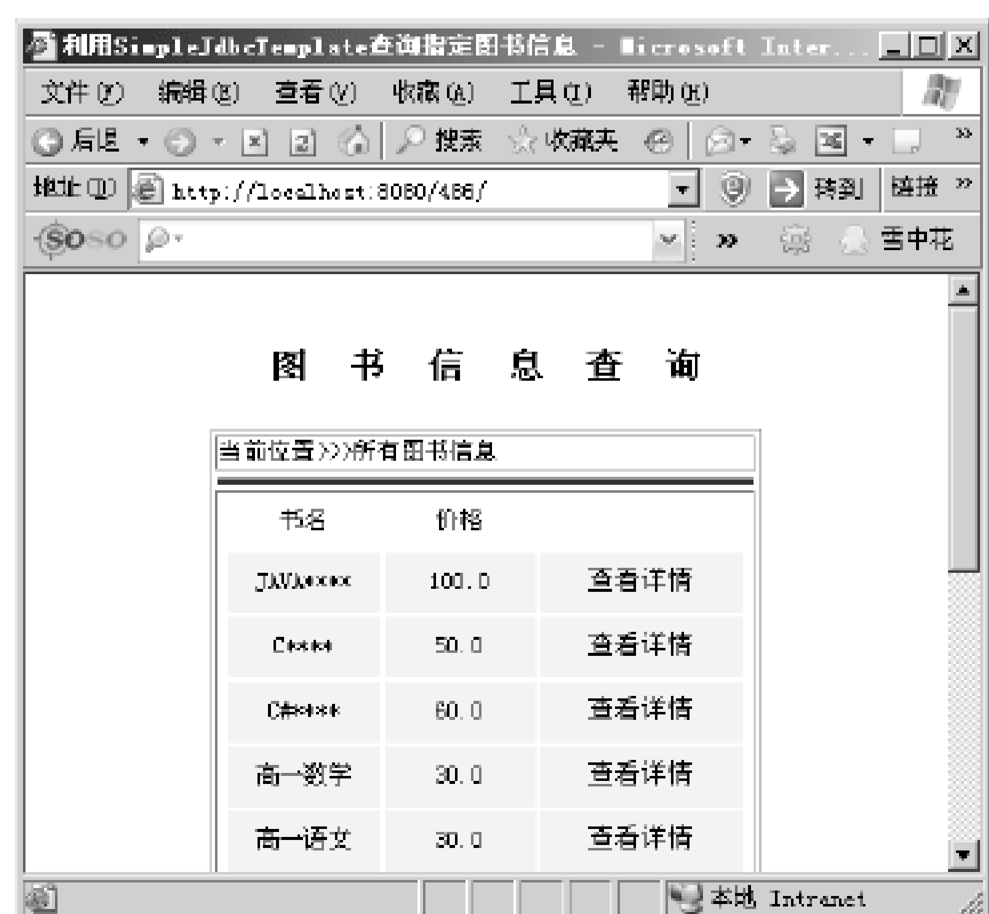


图 18.32　所有图书信息



图 18.33　图书详细信息

关键技术

SimpleJdbcTemplate 类的 queryForList()方法是 Spring 封装好的查询方法。调用该方法时，需要传递两个参

数，分别为 SQL 语句和 SqlParameterSource，返回类型为 List。代码如下：

```
simpleJdbcTemplate.getNamedParameterJdbcOperations().queryForList(String sql,SqlParameterSource param);
```

（1）创建封装图书信息的 JavaBean 类 Books，关键代码如下：

```
public class Books {
    private int id;                         //图书编号
    private String name;                    //图书名称
    private float total;                    //图书价格
    private String species;                 //图书种类
    ……                                      //省略的 getter 和 setter 方法
}
```

（2）创建操作数据库的 Dao 类 BookDao，声明一个 simpleJdbcTemplate 类型属性，并添加 getter 和 setter 方法，用于将 simpleJdbcTemplate 注入到 BookDao；然后编写一个查询所有图书信息的方法和一个查询详细信息的方法。关键代码如下：

```
public class BookDao {
 private SimpleJdbcTemplate simpleJdbcTemplate;
……                                                                                    //省略的 getter 和 setter 方法
 public Iterator selectOne(int id){                                                   //查询一本图书的详细信息
        String sql="select * from tb_books where id="+id;
        Books book=new Books();
        SqlParameterSource param=new BeanPropertySqlParameterSource(book);
        Iterator it=simpleJdbcTemplate.getNamedParameterJdbcOperations().queryForList(sql, param).iterator();   //执行查询方法，把结果集转换成 Iterator
        return it;
 }
 public Iterator selectAll(){                                                         //查询所有图书的信息
        String sql="select id,name,total from tb_books";
        Books book=new Books();
        SqlParameterSource param=new BeanPropertySqlParameterSource(book);
        Iterator it=simpleJdbcTemplate.getNamedParameterJdbcOperations().queryForList(sql, param).iterator();   //执行查询方法，把结果集转换成 Iterator
        return it;
 }
}
```

（3）在 Spring 的 applicationContext.xml 文件中，首先配置 DataSource 数据源；然后配置 simpleJdbcTemplate，将数据源对象注入 simpleJdbcTemplate 中；最后配置 BookDao，并将 simpleJdbcTemplate 注入 BookDao。关键代码如下：

```
<!-- 配置 simpleJdbcTemplate -->
<bean id="simpleJdbcTemplate" class="org.springframework.jdbc.core.simple.SimpleJdbcTemplate">
        <constructor-arg><ref bean="dataSource" /></constructor-arg>
</bean>
<!-- 配置 Dao -->
<bean id="bookDao" class="com.dao.BookDao">
        <property name="simpleJdbcTemplate">
                <ref local="simpleJdbcTemplate"/>
        </property>
</bean>
```

（4）创建显示所有图书信息的表单页 index.jsp。关键代码如下：

```
<%
        Resource resource = new ClassPathResource("applicationContext.xml");        //装载配置文件
        BeanFactory factory = new XmlBeanFactory(resource);
        BookDao dao = (BookDao)factory.getBean("bookDao");                          //获取 Dao 对象
        Iterator it=dao.selectAll();                                                //调用方法，查询员工信息
        while(it.hasNext()) {
                Map map = (Map) it.next();
%>
         <tr align=center bgcolor="#f1f3f5">
                <td height="30" align="center"><%=map.get("name")%></td>
                 <td><%=map.get("total") %></td>
                <td><a href="getone.jsp?sid=<%=map.get("id")%>">查看详情</a> </td>
               </tr>
      <% }   %>
```

（5）创建显示指定图书详细信息的表单页 getone.jsp，具体代码参见配书光盘。

秘笈心法

心法领悟 486：通过超链接传递参数。

本实例就是通过超链接传递参数，以获取想要查看详细图书信息的 id，再通过 id 的值获取图书的详细信息。超链接通过“?”传递参数；如果想同时传递多个参数，需用“&”传递参数。代码如下：

```
<a href="xxx.jsp?username=aaa">跳转到 xxx 页面</a>                    //传递一个参数
<a href="xxx.jsp?username=aaa&password=aaa">跳转到 xxx 页面</a>       //传递多个参数
```

实例 487　在 Spring 中配置 DBCP 数据库连接池　高级

光盘位置：光盘\MR\18\487

实用指数：★★★

实例说明

在 Spring 中已经集成了 DBCP 数据库连接池，用户只需进行简单的配置即可使用数据库连接池。本例将介绍如何在 Spring 中配置 DBCP 数据库连接池，将学生数据添加到数据库，如图 18.34 和图 18.35 所示。

图 18.34　输入学生信息

StuId	StuName	StuClass
1	张三	一年五班
2	李四	二年一班
3	王五	二年一班
4	王*	二年一班
5	赵*	三年一班

图 18.35　查询是否插入成功

关键技术

由于 JdbcDaoSupport 类中包含一个 JdbcTemplate 实例变量，并且已经开放了 DataSource 接口，因此只需简单地扩展 JdbcDaoSupport 即可定义用户自己的 Dao 类。代码如下：

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
```

设计过程

（1）创建名为 Students 的 JavaBean 类，用于封装学生信息。关键代码如下：

```
public class Students {
    private int StuId;                    //学生 id
    private String StuName;               //学生姓名
    private String StuClass;              //学生班级
    ……                                    //省略的 setter 和 getter 方法
}
```

（2）创建操作学生信息的接口 StudentDao，并定义添加商品信息的 addStudent()方法，参数类型为 Students 实体对象。代码如下：

```
public interface StudentDao {
 public void addStudents(Students student);
}
```

（3）编写 StudentDAO 接口的实现类——StudentDaoImpl 类，并在该类中实现接口中定义的方法 addStudents()，通过此方法访问数据库。关键代码如下：

```
public void addStudents(Students student) {
        Connection conn=null;
        PreparedStatement stmt=null;
        try{
```

```
                conn = dataSource.getConnection();
                String sql = "insert into tb_student(StuName,StuClass) values(?,?)";
                stmt = conn.prepareStatement(sql);
                stmt.setString(1, student.getStuName());
                stmt.setString(2, student.getStuClass());
                stmt.executeUpdate();
        }catch(Exception ex){
                ex.printStackTrace();
        }
        finally{
                try {
                        stmt.close();
                        conn.close();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }
}
```

（4）创建填写学生信息的表单页 index.jsp，具体代码参见配书光盘。

（5）创建处理表单数据的 save.jsp 页。代码如下：

```
<%
request.setCharacterEncoding("GBK");
String StuName = request.getParameter("StuName");                    //获取表单数据
String StuClass = request.getParameter("StuClass");
Students student = new Students();                                   //创建封装学生信息的 JavaBen
student.setStuName(StuName);
student.setStuClass(StuClass);
Resource resource = new ClassPathResource("applicationContext.xml");
BeanFactory factory = new XmlBeanFactory(resource);
StudentDaoImpl dao = (StudentDaoImpl)factory.getBean("StudentDao");  //获取 Dao 对象
dao.addStudents(student);                                            //执行添加方法
out.println("<script type='text/javascript'> alert('添加成功！ ');window.location.href='index.jsp'</script>");
%>
```

秘笈心法

心法领悟 487：JdbcTemplate 类对数据库的操作。

- void execute(String sql)：主要用于执行 DLL 语句。
- List queryForList(String sql)：执行 SQL 查询，将每行记录包装成 List 对象，返回这些 List 组成的集合。
- int update(String sql,Object[] args)：执行 SQL 更新，SQL 允许使用参数。

实例 488　在 Spring 中使用占位符配置数据源

光盘位置：光盘\MR\18\488

高级　实用指数：★★★

实例说明

本实例通过在 Spring 的配置文件中使用占位符来配置数据源，之后测试将学生信息保存到数据库中。运行程序，在如图 18.36 所示页面中输入学生姓名和班级，然后单击“添加到数据库”按钮，学生信息就会被保存到数据库中。

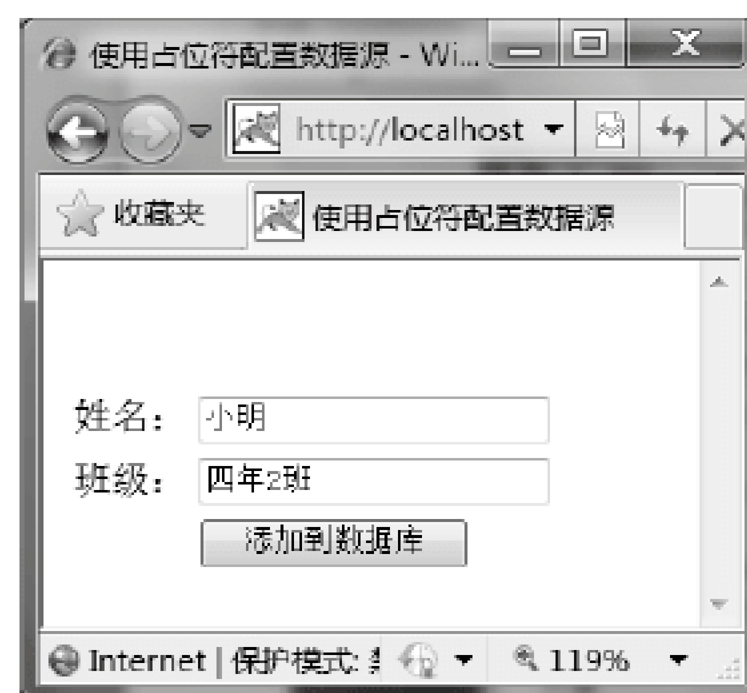


图 18.36　使用占位符配置数据源后添加学生信息

Spring 提供了一个 BeanFactoryPostProcessor 的实现类，即 PropertyPlaceholderConfigurer。该类允许在 XML 配置文件中使用占位符（Place Holder），并且将这些占位符所代表的资源单独配置到 properties 文件中加载。占位符类似于 JSP 中的 EL 表

达式，使用${}的格式，${}大括号内指定的值就是 properties 文件中的键。

例如，本实例要实现通过占位符配置数据源，首先应在配置文件中定义数据源的相关配置。代码如下：

```
jdbc.driver = com.mysql.jdbc.Driver
```

接下来，在 Spring 的配置文件中通过占位符配置代码数据源。代码如下：

```
<bean class="org.springframework.beans.factory.config.PreferencesPlaceholderConfigurer">
        <property name="locations">
                <value>dbcon.properties</value>
        </property>
</bean>
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
        <property name="driverClassName">
                <value>${jdbc.driver}</value>
        </property>
        ……
</bean>
```

（1）创建配置数据库连接的 properties 文件，配置代码如下：

```
jdbc.url=jdbc:mysql://localhost:3306/db_database18
jdbc.driver=com.mysql.jdbc.Driver
jdbc.username=root
jdbc.password=111
```

（2）在 Spring 的 XML 配置文件中，配置 PropertyPlaceholderConfigurer，通过占位符的形式加载 properties 文件的配置，然后通过占位符配置数据源。代码如下：

```
<bean class="org.springframework.beans.factory.config.PreferencesPlaceholderConfigurer">
        <property name="locations">
                <value>dbcon.properties</value>
        </property>
</bean>
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
        <property name="driverClassName">
                <value>${jdbc.driver}</value>
        </property>
        <property name="url" >
                <value>${jdbc.url}</value>
        </property>
        <property name="username">
                <value>${jdbc.username}</value>
        </property>
        <property name="password">
                <value>${jdbc.password}</value>
        </property>
</bean>
<bean id="StudentDao" class="com.dao.impl.StudentDaoImpl">
        <property name="dataSource">
                <ref local="dataSource"/>
        </property>
</bean>
```

（3）在学生信息的保存页面 save.jsp 中，通过 ApplicationContext 加载配置文件，并完成数据添加操作。代码如下：

```
<%
request.setCharacterEncoding("GBK");
String StuName = request.getParameter("StuName");
String StuClass = request.getParameter("StuClass");
Students student = new Students();
student.setStuName(StuName);
student.setStuClass(StuClass);
ApplicationContext context=new ClassPathXmlApplicationContext("applicationContext.xml");   //获取 ApplicationContext 容器
StudentDaoImpl dao = (StudentDaoImpl) context.getBean("StudentDao");                       //指定 Bean 的名称来取得 Bean 实例
dao.addStudents(student);
out.println("<script type='text/javascript'> alert('添加成功！ ');window.location.href='index.jsp'</script>");
%>
```

秘笈心法

心法领悟 488：ApplicationContext 加载 BeanFactoryPostProcessor。

BeanFactoryPostProcessor 是 Spring 提供的一种容器的扩展机制，该机制允许在容器实例化相应对象之前，对注入到容器的 BeanDefinition 所保存的信息进行相应的修改。本实例中应用的 PropertyPlaceholderConfigurer 类是 BeanFactoryPostProcessor 的一个实现类。相对于 BeanFactory，应用 ApplicationContext 来加载并应用 BeanFactoryPostProcessor 比较简单。因为 ApplicationContext 会自动识别配置文件中的 BeanFactoryPostProcessor 并应用它，所以只需将相应的 BeanFactoryPostProcessor 实现类添加到配置文件中即可，其他由 ApplicationContext 自动处理。

实例 489 使用 destroy-method 处理数据源

光盘位置：光盘\MR\18\489

高级

实用指数：★★★

实例说明

本实例通过在 Spring 的配置文件中使用 destroy-method 来释放注册的数据库连接池资源。运行程序，在如图 18.37 所示页面中输入学生姓名和班级，然后单击“添加到数据库”按钮，学生信息就会被保存到数据库中。

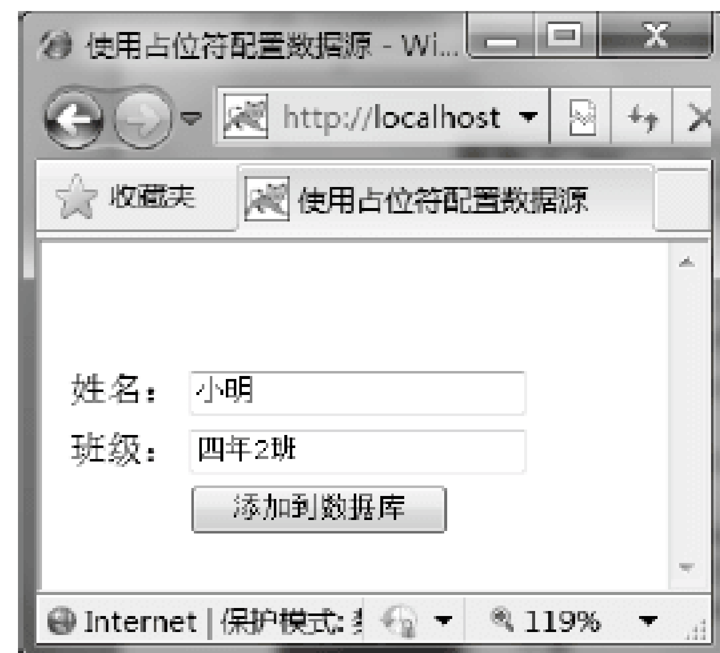


图 18.37　添加学生信息

关键技术

在 Spring 中，与 init-method 用于对象的自定义初始化相对应，destroy-method 为对象提供了执行自定义销毁的机会。该功能最常见的使用场景就是在 Spring 容器中注册数据库连接池，在系统退出后，连接池应该关闭，以释放相应资源。

在 Spring 的 XML 配置文件中，destroy-method 是<bean>元素的属性，当该属性值为 close 时，在系统关闭时就会及时地释放连接池的资源。

在 Spring 的 XML 配置文件中，在配置数据库连接池的<bean>元素中将 destroy-method 属性设置为 close，以便关闭系统时释放连接池资源。代码如下：

```
<bean class="org.springframework.beans.factory.config.PreferencesPlaceholderConfigurer">
        <property name="locations">
                <value>dbcon.properties</value>
        </property>
</bean>
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
        <property name="driverClassName">
                <value>${jdbc.driver}</value>
        </property>
        <property name="url" >
                <value>${jdbc.url}</value>
        </property>
        <property name="username">
                <value>${jdbc.username}</value>
        </property>
        <property name="password">
                <value>${jdbc.password}</value>
        </property>
        <property name="testOnBorrow">
                <value>true</value>
        </property>
        <property name="testOnReturn">
```

```
                <value>true</value>
        </property>
        <property name="testWhileIdle">
                <value>true</value>
        </property>
        <property name="minEvictableIdleTimeMillis">
                <value>180000</value>
        </property>
        <property name="timeBetweenEvictionRunsMillis">
                <value>360000</value>
        </property>
        <property name="maxActive">
                <value>100</value>
        </property>
</bean>
<bean id="StudentDao" class="com.dao.impl.StudentDaoImpl">
        <property name="dataSource">
                <ref local="dataSource"/>
        </property>
</bean>
```

秘笈心法

心法领悟 489：destroy-method 是如何销毁对象的。

当所有的参数方法都已经设置、注入或调用完成之后，容器将检查 singleton 类型的 Bean 实例，看其对应的 Bean 定义是否通过<bean>的 destroy-method 属性指定了自定义的对象销毁方法，如果是，就会为该实例注册一个用于对象销毁的回调（Callback），以便在这些 singleton 类型的对象实例销毁之前，执行销毁逻辑。

实例 490　Spring 分页显示图书信息

光盘位置：光盘\MR\18\490

高级

实用指数：★★★

实例说明

如果数据信息量很大，就要使用分页的方法来显示数据。Spring 提供了一个自动将数据进行分页的类 PagedListHolder，本实例将通过该类来实现分页显示数据。运行本实例，即可看到数据以分页的形式显示，如图 18.38 所示。单击“上一页”、“下一页”超链接，可以显示出上一页或下一页中的数据。



图 18.38　Spring 分页显示图书信息

关键技术

本实例中，在分页显示数据时主要用到了 PagedListHolder 类。该类是 Spring 为实现分页功能提供的类，其中封装了一些分页常会用到的功能，分别介绍如下。

- setPageSize()：设置每页记录数。
- getPageList()：获取当前页中的内容。
- isFirstPage()：判断是否为第一页。
- isLastPage()：判断是否为最后一页。
- nextPage()：下一页。
- getPageCount()：获取总页数。
- setPage()：选定指定的页数。

（1）编写 BookDao.java 类文件，该类继承自 JdbcDaoSupport 类，是用于对数据库进行操作的 Dao 类。关

键代码如下：

```
public class BookDao extends JdbcDaoSupport {
 public List findPaeg(){
        List list = getJdbcTemplate().queryForList("select * from tb_books");        //执行数据库查询 SQL 语句
                return list;                                                    //返回 List 列表
 }
}
```

（2）编写控制器类 ShowInfoController.java 文件，该类继承自 Controller 类。在该类中调用查询数据库的方法获取查询结果，将查询到的结果保存到 Session 对象中。关键代码如下：

```
public class ShowInfoController implements Controller{
 private BookDao bookDao;                                   //操作数据库的 Dao 类
 public BookDao getbookDao() {
        return bookDao;
 }
 public void setBookDao(BookDao bookDao) {
        this.bookDao = bookDao;
 }
 public ModelAndView handleRequest(HttpServletRequest request,
                HttpServletResponse response) throws Exception {
        List list =bookDao.findPaeg();                       //调用查询数据库的方法，获取查询结果
        request.getSession().setAttribute("bookList", list);  //将结果存入 Session 对象
        return new ModelAndView("show");                     //回到视图页面
 }
}
```

（3）编写 applicationContext.xml 配置文件。关键代码如下：

```
<!-- 定义视图分解器 -->
<bean id="viewResolver"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <!-- 设置前缀，即视图所在的路径 -->
        <property name="prefix" value="/" />
        <!-- 设置后缀，即视图的扩展名 -->
        <property name="suffix" value=".jsp" />
</bean>
<bean id="bookDao" class="com.dao.BookDao">
        <property name="dataSource" ref="dataSource" />
</bean>
<bean name="/showInfo.do" class="com.controller.ShowInfoController">
        <property name="bookDao" ref="bookDao" />
</bean>
```

（4）编写 show.jsp 页面文件，在该文件中首先从 Session 对象中取出查询数据库获得的 List 列表，然后使用 PagedListHolder 类对 List 列表进行分页操作。关键代码如下：

```
<table border="1" width="450" cellpadding="1" cellspacing="1" bordercolor="#FFFFFF" bgcolor="#009B89" style="position: relative; top:-10px;
left:260px;"  >
                <tr>
                        <td width="41" height="23" align="center" bordercolor="#FFFFFF"><span class="STYLE1">图书编号</span></td>
                        <td width="128" height="23" align="center" bordercolor="#FFFFFF"><span class="STYLE1">图书名称</span></td>
                        <td width="63" height="23" align="center" bordercolor="#FFFFFF"><span class="STYLE1">图书价格</span></td>
                </tr>
                <%
                int cPage = 0;
                int pageSize = 5;
                if(request.getParameter("cPage")!=null){
                        cPage = Integer.parseInt(request.getParameter("cPage"));
                }
                PagedListHolder plist = new PagedListHolder();
                plist.setPageSize(pageSize);
                plist.setSource((List)session.getAttribute("bookList"));
                plist.setPage(cPage);
                List list = plist.getPageList();
                Iterator it = list.iterator();
                while (it.hasNext()) {
                        Map userMap = (Map) it.next();
        %>
                <tr>
                        <td height="23" align="center" bordercolor="#FFFFFF" bgcolor="#FFFFFF"><%=userMap.get("id") %></td>
```

```
				<td height="23" bordercolor="#FFFFFF" bgcolor="#FFFFFF"><%=userMap.get("name") %></td>
				<td height="23" bordercolor="#FFFFFF" bgcolor="#FFFFFF"><%=userMap.get("total") %></td>
			</tr>
		<%
			}
		%>
		<tr>
		<td height="23" colspan="5" align="center" bordercolor="#FFFFFF" bgcolor="#FFFFFF">总页数：<%= plist.getPageCount() %> <%= plist.isFirstPage()?"":"<a href='show.jsp?cPage="+(cPage-1)+"'>上一页</a>" %> 当前第<%=cPage+1%>页 <%= plist.isLastPage()?"":"<a href='show.jsp?cPage="+(cPage+1)+"'>下一页</a>" %></td>
		</tr>
	</table>
```

（5）在 web.xml 文件中配置 Spring 总控器。关键代码如下：

```
<servlet>
	<servlet-name>dispatcherServlet</servlet-name>
	<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
	<init-param>
		<param-name>contextConfigLocation</param-name>
		<param-value>/WEB-INF/applicationContext.xml</param-value>
	</init-param>
	<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
	<servlet-name>dispatcherServlet</servlet-name>
	<url-pattern>*.do</url-pattern>
</servlet-mapping>
```

秘笈心法

心法领悟 490：request.setAttribute()和 request.getSession().setAttribute()的生命周期。

request.setAttribute()的生命周期是 request 级别，只能在当前请求中访问到，其他请求无法访问；而 request.getSession().setAttribute()的生命周期是 session 级别，只要当前会话不过期，任何地方都可以访问到。

实例 491　整合 Spring 和 Hibernate 添加员工信息

光盘位置：光盘\MR\18\491

高级

实用指数：★★★

实例说明

Spring 中整合了 Hibernate 框架来完成数据的持久化，在集成环境下使用 Hibernate 可以简化程序编写的流程。本实例利用 Spring 整合 Hibernate 实现向员工信息表中添加数据。运行本实例，在如图 18.39 所示页面中输入员工信息，单击“添加到数据库”按钮，即可将相应信息添加到数据库，如图 18.40 所示。

图 18.39　输入员工信息

Resultset 1

id	name	sex	age	dept	duty	telephone
1	孙*	男	23	销售部	员工	138****7777
2	王*	女	35	销售部	经理	138****8888

图 18.40　查询数据库信息

关键技术

在 Spring 框架的 HibernateDaoSupport 类中，向数据表中插入数据的方法为 save()，该方法的参数的类型为

Object 对象。

```
this.getHibernateTemplate().save(emp);
```

（1）定义员工信息的实体 JavaBean 对象。关键代码如下：

```
public class Employee {
 private int id;                  //编号
 private String name;             //姓名
 private String sex;              //性别
 private int age;                 //年龄
 private String dept;             //部门
 private String duty;             //职务
 private String telephone;        //联系电话
 ……                               //省略的 getter 和 setter 方法
}
```

（2）编写实体对象与数据表对应的文件 Employee.hbm.xml。关键代码如下：

```
<!-- Employee 信息字段配置信息 -->
 <hibernate-mapping>
        <class name="com.lh.bean.Employee" table="tb_employee2">
                <!-- id 值 -->
        <id name="id" column="id" type="int">
                <generator class="native"/>
        </id>
        <!-- 姓名 -->
        <property name="name" type="string" length="45">
                <column name="name"/>
        </property>
        <!-- 年龄 -->
        <property name="age" type="int">
                <column name="age"/>
        </property>
        ……<!--省略了其他字段的配置-->
        </class>
 </hibernate-mapping>
```

（3）创建操作数据库的 DAO 类 EmpDao，该类继承自 HibernateDaoSupport。关键代码如下：

```
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import com.lh.bean.Employee;
public class EmpDao extends HibernateDaoSupport     {
 public void addEmp(Employee emp){
        this.getHibernateTemplate().save(emp);          //保存员工对象
 }
}
```

（4）编写 Spring 配置文件 applicationContext.xml，在该配置文件中设置 Bean 之间的依赖关系。关键代码如下：

```
<!-- 定义 Hibernate 的 sessionFactory -->
<bean id="sessionFactory"
        class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
        <property name="dataSource">
                <ref bean="dataSource" />
        </property>
        <property name="hibernateProperties">
                <props>
                        <!-- 数据库连接方言 -->
                        <prop key="dialect">org.hibernate.dialect.SQLServerDialect</prop>
                        <!-- 在控制台输出 SQL 语句 -->
                        <prop key="hibernate.show_sql">true</prop>
                        <!-- 格式化控制台输出的 SQL 语句 -->
                        <prop key="hibernate.format_sql">true</prop>
                </props>
        </property>
        <!--Hibernate 映射文件 -->
        <property name="mappingResources">
                <list>
                        <value>com/lh/bean/Employee.hbm.xml</value>
```

```
            </list>
        </property>
</bean>
<!-- 注入 SessionFactory -->
<bean id="empDao" class="com.lh.dao.EmpDao">
        <property name="sessionFactory">
                <ref local="sessionFactory" />
        </property>
</bean>
```

（5）编写用于录入员工信息的表单页 index.jsp 和处理表单请求的页面 save.jsp，具体代码参见配书光盘。

秘笈心法

心法领悟 491：generator 属性。

generator 属性配置的是主键生成方式。本实例中 class 的值为 native，表示由 Hibernate 根据使用的数据库自行判断采用 identity（采用数据库提供的主键生成机制）、hilo（通过 hi/lo 算法实现主键生成机制）、sequence（采用数据库提供的 sequence 机制生成主键）中的一种作为主键生成方式。

实例 492　整合 Spring 和 Hibernate 批量添加用户信息

光盘位置：光盘\MR\18\492

高级　实用指数：★★★

实例说明

本实例将结合 Spring 和 Hibernate 框架实现数据的批量添加功能。程序运行后，在如图 18.41 所示页面中填写用户信息，在“添加信息数量”文本框中输入插入的记录数量，然后单击“添加”按钮，即可在数据库中自动插入指定数量的记录。



图 18.41　填写用户信息

关键技术

本实例利用循环添加信息数量的值，应用 save()方法完成批量添加。代码如下：

```
for(int i = 0; i < count ; i++){                    //批量执行添加方法
        User userVO = new User();                   //实例化对象
        userVO.setName(name+i);                     //设置用户名
        userVO.setBusiness(business);               //设置职务
        userVO.setAddTime(new Date());              //设置添加时间
        template.save(userVO);                      //执行添加方法
    }
```

设计过程

（1）定义用户信息的实体 JavaBean 对象。关键代码如下：

```
public class User{
 private Integer id;                //id 值
 private String name;               //用户名称
 private String business;           //职务
 private Date addTime;              //添加时间
```

```
}
```

（2）编写实体对象与数据表对应的文件 User.hbm.xml。关键代码如下：

```
<hibernate-mapping>
<class name="com.mr.user.User" table="tb_user">
	<!-- id 值 -->
		<id name="id" column="id" type="int">
		<generator class="native"/>
	</id>
	<!-- 名称 -->
	<property name="name" type="string" length="45">
		<column name="name"/>
	</property>
	<!-- 职务 -->
	<property name="business" type="string" length="45">
		<column name="business"/>
	</property>
	<!-- 信息添加时间 -->
	<property name="addTime" type="java.util.Date">
		<column name="addTime"/>
	</property>
</class>
</hibernate-mapping>
```

（3）创建操作数据库的 DAO 类 DAOSupport，该类继承自 HibernateDaoSupport。关键代码如下：

```
public class DAOSupport extends HibernateDaoSupport {
public void InsertPatchInfo(int count,HttpServletRequest request,HttpServletResponse response){
	String name = request.getParameter("name");                          //用户名称
	String business = request.getParameter("business");                  //职务
	HibernateTemplate template = this.getHibernateTemplate();            //实例化 Hibernate 模板
	for(int i = 0; i < count ; i++){                                     //批量执行添加方法
		User userVO = new User();                                        //实例化对象
		userVO.setName(name+i);                                          //设置用户名
		userVO.setBusiness(business);                                    //设置职务
		userVO.setAddTime(new Date());                                   //设置添加时间
		template.save(userVO);                                           //执行添加方法
	}
}
}
```

（4）创建批量添加信息的控制器 AddUser 类，该类继承自 AbstractController。关键代码如下：

```
public class AddUser extends AbstractController {
private DAOSupport dst;                                                  //注入 DAOSupport
……                                                                       //省略的 setter 和 getter 方法
protected ModelAndView handleRequestInternal(HttpServletRequest request,
		HttpServletResponse response) throws Exception {
	request.setCharacterEncoding("UTF-8");                               //添加信息的条数
	int count = Integer.parseInt(request.getParameter("count"));
	dst.InsertPatchInfo(count,request,response);                         //执行封装的批量添加方法
	Map map = new HashMap();                                             //定义映射
	map.put("succ", "信息添加成功，共添加"+count+"条信息！");               //设置添加成功时的提示信息
	return new ModelAndView("index",map);
}
}
```

（5）编写 Spring 配置文件 applicationContext.xml，在该配置文件中设置 Bean 之间的依赖关系。关键代码如下：

```
<!-- 定义 Hibernate 的 sessionFactory -->
<bean id="sessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
	<property name="dataSource">
		<ref bean="dataSource" />
	</property>
	<property name="hibernateProperties">
		<props>
			<!-- 数据库连接方言 -->
			<prop key="dialect">org.hibernate.dialect.SQLServerDialect</prop>
			<!-- 在控制台输出 SQL 语句 -->
			<prop key="hibernate.show_sql">true</prop>
			<!-- 格式化控制台输出的 SQL 语句 -->
```

```
                <prop key="hibernate.format_sql">true</prop>
            </props>
        </property>
        <!--hibernate 映射文件 -->
        <property name="mappingResources">
            <list>
                <value>com/mr/user/User.hbm.xml</value>
            </list>
        </property>
</bean>
<!-- 为 DAOSupport 注入 sessionFactory -->
<bean id="dao" class="com.mr.dao.DAOSupport">
        <property name="sessionFactory">
            <ref local="sessionFactory"/>
        </property>
</bean>
<!--定义控制器转发视图类 -->
<bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix">
            <value>/</value>
        </property>
        <property name="suffix">
            <value>.jsp</value>
        </property>
</bean>
<!-- 映射的 do -->
<bean name="/save.do" class="com.mr.main.AddUser">
        <property name="dst">
            <ref local="dao"/>
        </property>
</bean>
```

（6）编写用于输入用户信息的表单页 index.jsp，具体代码参见配书光盘。

■ 秘笈心法

心法领悟 492：配置 SessionFactory。

LocalSessionFactoryBean 是处理 XML 方式的 Bean，利用 mappingResources 配置实体类映射文件，再利用 hibernateProperties 配置相关的属性。

18.5　在 Spring 中生成非 HTML 输出

实例 493	利用 Spring 将学生信息导出到 Excel 工作表 光盘位置：光盘\MR\18\493	高级 实用指数：★★★★★

■

本实例主要介绍如何利用 Spring 框架将学生信息导入 Excel 文件，并可对该文件进行修改操作，同时还可将生成的 Excel 文件保存到本地磁盘中。程序运行后，单击“提交”按钮，在 JSP 页面中将生成 Excel 文件，如图 18.42 所示。

A	B	C	D	E	F	G	H
	初中三班 学生详细信息						
学生姓名	性别	身份证号	出生日期	政治面貌	家庭电话	家庭地址	健康状况
陈*	女	220********	1991-11-11	团员	8********	吉林省长春市	优
钱*	男	220********	1991-11-11	团员	8********	吉林省长春市	优
周*	女	220********	1991-11-11	**	8********	吉林省长春市	优

图 18.42　生成的 Excel 文件

关键技术

本实例的实现用到了 Apache 的开源免费插件 poi 3.0，通过这个类包来完成 Excel 文件的制作。

- HSSFSheet.java：生成一个 sheet 工作区，通过 HSSFWorkbook.java 调用 create()方法进行创建。代码如下：

```
HSSFSheet wbsheet = workbook.createSheet(title);
```

- HSSFRow.java：生成一行数据，通过 HSSFSheet 类的对象 sheet 创建工作区。代码如下：

```
HSSFRow titlename = wbsheet.createRow(0);
          HSSFCell celltitle = titlename.createCell((short)1);
 celltitle.setCellValue(title);
```


（1）创建 UserXslController.java 类文件，实现接口类 Controller 响应的用户请求操作。其关键代码如下：

```
public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse reponse) throws Exception {
      String name = request.getParameter("selectname");                    //从 JSP 页面中取得班级名称
      String classname = new String(name.getBytes("ISO8859_1"),"GBK");
      DriverManagerDataSource dmds = new DriverManagerDataSource();
      dmds.setDriverClassName("com.mysql.jdbc.Driver");
      dmds.setUrl("jdbc:mysql://localhost:3306/db_database18");
      dmds.setUsername("root");
      dmds.setPassword("111");
      JdbcTemplate jtl = new JdbcTemplate(dmds);                          //取得数据库连接
      List listcontent = jtl.queryForList("SELECT stu_id, name, sex, age, sfzhm,csrq,zzmm, jtdh,jtdz,jkzk FROM tb_stuinfo where classid = '" + 
classname + "'");
      Map map = new HashMap();
      map.put("title", classname + " 学生详细信息");
      map.put("listdata", listcontent);                                    //把查询的数据存放在 Map 对象中
      return new ModelAndView("customxslview", map);
}
```

（2）创建UserXslView.java类文件，继承AbstractExcelView，并且生成buildExcelDocument()方法，以生成Excel文件。该类的完整代码如下：

```
protected void buildExcelDocument(Map model, HSSFWorkbook workbook,
HttpServletRequest request, HttpServletResponse response) throws Exception {
```

首先，通过Map对象获得传递的数据，创建工作区和标题并设置其风格。代码如下：

```
List listdata = (List)model.get("listdata");              //创建工作表
String title = model.get("title").toString();             //创建标题
HSSFSheet wbsheet = workbook.createSheet(title);
HSSFRow titlename = wbsheet.createRow(0);
HSSFCell celltitle = titlename.createCell((short)1);
HSSFCellStyle celltitlestyle = workbook.createCellStyle();
 HSSFFont cellfont = workbook.createFont();
cellfont.setFontHeightInPoints((short)8);
cellfont.setFontHeight((short)HSSFFont.BOLDWEIGHT_NORMAL);
cellfont.setColor((short)(HSSFFont.COLOR_RED));
celltitlestyle.setFont(cellfont);
celltitle.setCellStyle(celltitlestyle);
celltitle.setCellValue(title);
```

其次，通过循环语句创建工作表的标题并且设置其相应的风格属性。代码如下：

```
String titles[] = {"学生姓名","性别","身份证号","出生日期","政治面貌","家庭电话","家庭地址","健康状况"};
   HSSFRow dataTitleRow = wbsheet.createRow((short) 1);
   HSSFCellStyle celltbnamestyle = workbook.createCellStyle();
   celltbnamestyle.setAlignment((short)HSSFCellStyle.ALIGN_CENTER);
   HSSFFont celltbnamefont = workbook.createFont();
   celltbnamefont.setFontHeightInPoints((short)10);
   celltbnamefont.setColor((short)(HSSFFont.COLOR_RED));
   celltbnamestyle.setFont(celltbnamefont);
   celltbnamestyle.setWrapText(true);
   for (int i = 0; i < titles.length; i++) {
     HSSFCell cell = dataTitleRow.createCell((short)i);
     if(i == 3 || i == 6 || i == 2) {
       wbsheet.setColumnWidth((short)i,(short)5335);
    }else{
        wbsheet.setColumnWidth((short)i,(short)3335);
```

```
        }
        cell.setCellValue(titles[i]);
        cell.setCellStyle(celltbnamestyle);
    }
}
```

最后，通过循环语句创建工作区中的数据单元格并且生成相应的数据。代码如下：

```
Object[] array = listdata.toArray();
HSSFCellStyle celldatastyle = workbook.createCellStyle();
HSSFDataFormat df = workbook.createDataFormat();
celldatastyle.setDataFormat(df.getFormat("yyyy-mm-dd"));
for(int i = 0 ; i < array.length ; i++){
 String tname[] = {"name","sex","sfzhm","csrq","zzmm","jtdh","jtdz","jkzk"};
 Map mapdata = (Map)array[i];
 HSSFRow dataRow = wbsheet.createRow((short)(i+2));
 for (int j = 0 ; j < tname.length; j++){
        HSSFCell cell = dataRow.createCell((short)j);
        String data=null;
        data = mapdata.get(tname[j]).toString();
        if(j==3){
                data = mapdata.get(tname[j]).toString();
                data = data.substring(0, 10);
        }else{
                data = mapdata.get(tname[j]).toString();
        }
        cell.setCellValue(data);
 }
}
```

（3）在 WEB-INF 文件夹下创建 bean_config.xml 文件，用来定义生成用户的控制器请求。其关键代码如下：

```
<beans>
<bean id="viewResolver"
class="org.springframework.web.servlet.view.BeanNameViewResolver"/> <bean id="customxslview" class="com.UserXslView"/>
<bean name="/userxls.do" class="com.UserXslController"/>
</beans>
```

（4）在web.xml文件中设置Servlet，完成用户的响应操作。其关键代码如下：

```
<servlet>
<servlet-name>dispatcherServlet</servlet-name>
<servlet-class>
org.springframework.web.servlet.DispatcherServlet
</servlet-class>
<init-param>
<param-name>contextConfigLocation</param-name>
<param-value>/WEB-INF/bean_config.xml</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>dispatcherServlet</servlet-name>
<url-pattern>*.do</url-pattern>
</servlet-mapping>
```

（5）定义一个用来供用户进行班级选择的index.jsp文件。其关键代码如下：

```
<form action="userxls.do" method="post">
  <table width="306" border="0" cellspacing="0" cellpadding="0">
<tr>
<td width="99"><span class="style1">请选择学生班级</span></td>
          <td width="90">
            <select name="selectname">
                <option value="初中一班">初中一班</option>
                <option value="初中二班">初中二班</option>
                <option value="初中三班">初中三班</option>
            </select>    </td>
           <td width="117"><input name="Submit" type="submit" value="提交">
           <input name="Submit2" type="reset" value="重置"></td>
        </tr>
  </table>
</form>
```

秘笈心法

心法领悟 493：Map map=new HashMap。

Map 是一个接口，接口是不能直接被实例化的。HashMap 是 Map 的一种实现，Map map=new HashMap()就是将 map 实例化成一个 HashMap。这样做的优点是调用者不需要知道 Map 具体的实现，Map 接口和具体实现的映射由 Java 完成。

实例 494　利用 Spring 将图书信息导出到 PDF 文件

光盘位置：光盘\MR\18\494

高级

实用指数：★★★★★

实例说明

本实例实现的是利用 Spring 生成 PDF 文件。运行程序后，单击“提交”按钮，在 JSP 页面中将生成 PDF 文件，如图 18.43 所示。

计算机类图书详细信息

图书编号	图书名称	图书价格
1	JAVA****	100.0
2	C****	50.0
3	C#****	60.0
9	计算机***	100.0
10	数据库***	100.0
11	Web***	100.0

版权所有:吉林省明日科技有限公司　公司网址:www.mykj.com

图 18.43　生成的 PDF 文件

关键技术

本实例利用开源插件 itext 将图书信息导出到 PDF 文件，itext 是一个开放源码的 Java 类库，可以生成包含一段文字、表格和图片的 PDF 文件。

（1）创建 UserPdfController.java 类控制器文件，继承 Controller 接口，并且在 handleRequest()方法中编写代码，以访问数据库中的数据并为数据对象赋值，所用类与实例 493 中的数据库类基本相同，读者可参考配书光盘中的源文件。

（2）创建 UserPdfView.java 类文件，继承 AbstractPdfView，在该类的 buildPdfDocument()方法中编写生成 PDF 文件的代码。其关键代码如下：

```
protected void buildPdfDocument(Map model, Document document,PdfWriter writer,HttpServletRequest arg1, HttpServletResponse arg2) throws Exception {
	BaseFont bfComic = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);
	Font font = new Font(bfComic, 16,Font.NORMAL);                //声明字体对象
	font = new Font(bfComic,16,1,new Color(255,0,0));             //设置字体
	String header = (String)model.get("header");                   //获取标题
	Paragraph headerParagraph = new Paragraph(header,font);
	headerParagraph.setAlignment(Paragraph.ALIGN_CENTER);         //设置格式
	document.add(headerParagraph);                                 //添加到 Document 对象中
	font = new Font(bfComic, 10, Font.NORMAL);                     //声明字体样式
	List content = (List)model.get("content");
	Object[] array = content.toArray();
	String title[] = {"图书编号","图书名称","图书价格"};
	Table t = new Table(title.length);                             //声明表格对象
```

```
        t.normalize();
        t.setAutoFillEmptyCells(true);
        t.setAlignment(Table.ALIGN_CENTER);                        //居中
        t.setWidth(108);                                           //设置宽度
        Phrase pgTitle;
        for(int i = 0 ; i < title.length ; i++){
                Cell celltitle = new Cell();                       //声明单元格
                celltitle.add(new Phrase(title[i],font));          //循环加入表头信息
            celltitle.setHorizontalAlignment(Element.ALIGN_CENTER);
            t.addCell(celltitle);
        }
        t.endHeaders();
        Paragraph contentParagraph = null;
        for(int i = 0 ; i < array.length ; i++){
                String tname[] = {"id","name","total"};
                Map mapdata = (Map)array[i];
                for (int j = 0 ; j < tname.length; j++){
                        Cell celldata = new Cell();                //声明单元格对象
                        String data=null;
                        if(j==3){
                                data = mapdata.get(tname[j]).toString();
                                data = data.substring(0, 10);
                        }else{
                                data = mapdata.get(tname[j]).toString();
                        }
                        Paragraph pgdata = new Paragraph(data,font);
                        celldata.setHorizontalAlignment(Element.ALIGN_CENTER);
                        celldata.addElement(pgdata);
                        t.addCell(celldata);
                }
        }
        document.add(t);
        String copyright = (String)model.get("copyright");
        Paragraph copyrightParagraph = new Paragraph(copyright,font);
        copyrightParagraph.setAlignment(Paragraph.ALIGN_BOTTOM);
        document.add(copyrightParagraph);
}
```

（3）在 WEB-INF 文件夹下创建一个 bean-config.xml 文件，设置控制器的信息。关键代码如下：

```
<bean id="viewResolver"
class="org.springframework.web.servlet.view.BeanNameViewResolver"/>
<bean id="customPdfView" class="com.UserPdfView"/>
<bean name="/userpdf.do" class="com.UserPdfController"/>
```

秘笈心法

心法领悟 494：<load-on-starup>1</load-on-startup>。

标记容器是否在启动时就加载这个配置的 Servlet：当值为 0 或大于 0 时，表示容器在应用程序启动时就加载这个 Servlet；当小于 0 或者没有指定时，则表示容器在该 Servlet 被选择时才加载这个 Servlet。正数的值越小，启动该 Servlet 的优先级别越高。

18.6　Spring 文件上传与国际化

实例 495	利用 Spring 实现文件的上传 光盘位置：光盘\MR\18\495	高级 实用指数：★★★★★

实例说明

Spring 提供了对 Jakarta Commons FileUpload 文件上传组件的支持，该组件在 Spring MVC 中实现了文件上传

功能。运行本实例，选择要上传的文件后单击“上传”按钮，即可将文件上传到服务器，如图 18.44 所示。

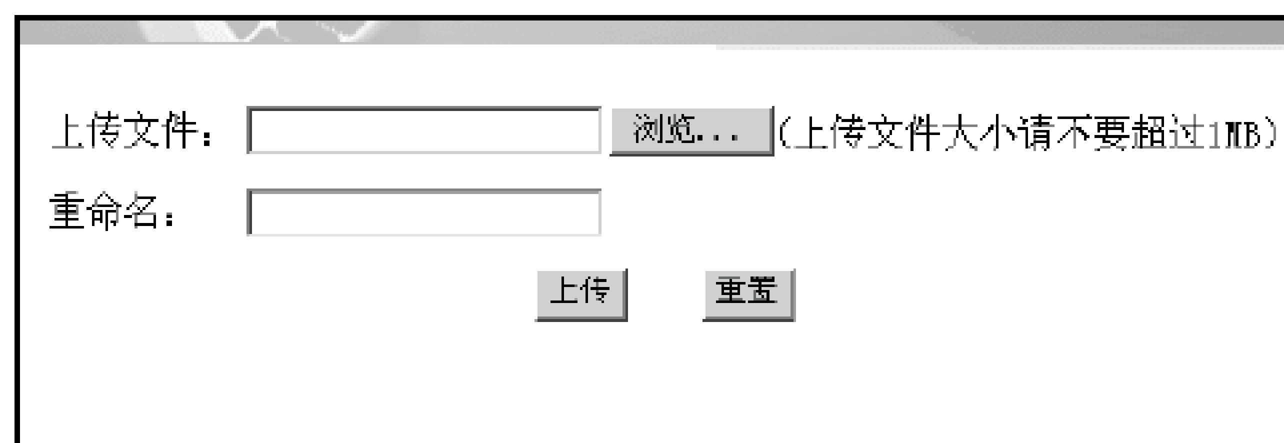


图 18.44　文件上传页面

关键技术

Spring 可以自动将要上传的文件装载到 MultipartFile 类型的属性中，MultipartFile 为操作上传文件提供了一些方法，分别介绍如下。

- byte[] getBytes()：获取文件数据。
- String getContentType()：获取文件的类型。
- InputStream getInputStream()：获取文件流。
- String getName()：获取表单对象中保存上传文件的属性名。
- String getOriginalFilename()：获取上传文件原名。
- long getSize()：获取文件大小，单位 byte。
- boolean isEmpty()：判断是否有上传文件。

（1）编写 MyFile.java 实体类，用于封装表单对象（使用 MultipartFile 类型的对象来封装文件）。关键代码如下：

```
public class MyFile {

 private String fileName;
 private MultipartFile myFile;
 ……                                    //省略的 setter 和 getter 方法
}
```

（2）编写处理文件上传的控制器 UploadController.java 类，该类只需要继承 SimpleFormController 类即可，主要用于获取表单中的数据并进行操作。关键代码如下：

```
protected ModelAndView onSubmit(HttpServletRequest request,
            HttpServletResponse response, Object command, BindException errors)
            throws Exception {
        MyFile myFile = (MyFile)command;
        String name = myFile.getMyFile().getOriginalFilename();                  //获取文件原名
                                                                                 //判断是否为文件重命名了
        if(myFile.getFileName()!=null&&!"".equals(myFile.getFileName())){
            name = myFile.getFileName();
        }
        String webRoot = WebUtils.getRealPath(request.getSession().getServletContext(), "/");   //获取项目的绝对路径
        File file = new File(webRoot+"\\uploadFile\\"+name);                     //保存文件，保存在 WEB-INF\uploadFile 文件夹中
        myFile.getMyFile().transferTo(file);
        return new ModelAndView("ok","path",file.toString());
}
```

（3）编写上传表单页面。关键代码如下：

```
<table style="position:relative; top:40px; left:290px">
<tr>
        <td height="32">上传文件：</td>
        <td height="32"><input type="file" name="myFile">(上传文件大小请不要超过 1MB)</td>
</tr>
        <tr>
        <td height="32">重命名：</td>
        <td height="32"><input type="text" name="fileName"></td>
```

```
</tr>
        <tr>
        <td height="32" colspan="2" align="center">
                <input type="submit" value="上传">   
                <input type="reset" value="重置">
        </td>
</tr>
</table>
```

（4）编写 ok.jsp 页面文件，显示出上传成功的文件在服务器中保存的绝对路径。关键代码如下：

```
<div style="position:relative; top:20px; left:280px">文件上传成功，保存在：<br><br>${path}<br>
        <br><a href="index.jsp">返回</a>
</div>
```

（5）编写 applicationContext.xml 配置文件。关键代码如下：

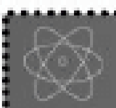
```
<!-- 定义视图分解器 -->
<bean id="viewResolver"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="viewClass">
                <value>org.springframework.web.servlet.view.InternalResourceView
                </value>
        </property>
        <!-- 设置前缀，即视图所在的路径 -->
        <property name="prefix" value="/" />
        <!-- 设置后缀，即视图的扩展名 -->
        <property name="suffix" value=".jsp" />
</bean>
<!-- 设置上传文件参数 -->
<bean id="multipartResolver"
        class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <!-- 编码格式，默认为 ISO-8859-1，这里必须要和页面中的编码相同 -->
        <property name="defaultEncoding" value="GBK" />
        <!-- 上传文件的大小限制，单位字节 -->
        <property name="maxUploadSize" value="1000000" />
        <!-- 上传时保存文件的临时目录，完成后文件会被自动删除 -->
        <property name="uploadTempDir" value="uploadFile/" />
</bean>
<!-- 声明控制器 -->
<bean name="/upLoad.do" class="com.jwy.controller.UploadController">
        <!-- 封装表单的类 -->
        <property name="commandClass" value="com.jwy.controller.MyFile"/>
        <!-- 上传文件的页面 -->
        <property name="formView" value="uploadFile"/>
        <!-- 上传成功后的页面 -->
        <property name="successView" value="ok"/>
</bean>
```

（6）在 web.xml 文件中配置 Spring 控制器。关键代码如下：

```
<servlet>
        <servlet-name>dispatcherServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>/WEB-INF/applicationContext.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
        <servlet-name>dispatcherServlet</servlet-name>
        <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

心法领悟 495：文件上传。

本实例实现的上传功能是把文件上传到服务器中该项目下的 uploadFile 文件夹中，而不是用户自己的 MyEclipse 项目下的 uploadFile 文件夹中。



实例 496 利用 Spring 实现用户登录页面的国际化

光盘位置：光盘\MR\18\496

高级

实用指数：★★★★★

实例说明

在 Spring MVC 中提供了多种实现国际化显示的方法。本实例将使用 ApplicationContext 容器来解析国际化资源，实现登录国际化，如图 18.45 所示。



图 18.45　程序初始化效果

关键技术

在 Spring 中可以通过 MessageSource 接口解析消息资源，其中定义了 ResourceBundleMessageSource 类的 JavaBean，该类是 MessageSource 接口的实现类，其 Bean 的名称必须定义为 messageSource，ApplicationContext 容器在初始化时会查找该名称的 JavaBean 来解析文本信息。

设计过程

（1）定制以下两个消息资源文件。

messages.properties 文件默认存储着简体中文消息资源，代码如下：

```
LoginPage.locale=\u4E2D\u56FD\u5927\u9646
LoginPage.loginName=\u7528\u6237\u540D
LoginPage.loginPass=\u5BC6\u7801
LoginPage.login=\u767B\u5F55
LoginPage.reset=\u9000\u51FA
```

messages_en_US.properties 文件存储着美式英语消息资源，代码如下：

```
LoginPage.locale=USA
LoginPage.loginName=User Name
LoginPage.loginPass=Password
LoginPage.login=Login
LoginPage.reset=Reset
```

（2）编写 Spring 的配置文件，其配置代码如下：

```
<!-- 配置 messageSource-->
<bean id="messageSource"
        class="org.springframework.context.support.ResourceBundleMessageSource">
        <property name="basename">
                <value>messages</value>
        </property>
</bean>
```

（3）在登录页面中加入如下代码：

```
<%
    ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
    Locale locale = null;
    if (request.getAttribute("language") == null) {
        locale = Locale.CHINA;
    } else {
        locale = (Locale) request.getAttribute("language");
    }
%>
```

心法领悟 496：资源文件。

消息的国际化需要编制多个对应特定语言的资源文件和一个必需的默认资源文件。例如，消息资源名为 messages，那么对应的默认资源文件就是 messages.properties，它是容器找不到对应语言的资源时默认读取的资源文件；而 messages_en_US.properties 对应美式英语资源；其他语言，如德语、墨西哥语等都有对应的资源文件。

第19章

Spring 的 Web MVC 框架

- Spring 的控制器
- 在线通讯录
- 图书信息管理

19.1　Spring 的控制器

实例 497　使用简单控制器获取表单数据

光盘位置：光盘\MR\19\497

高级　实用指数：★★★★

实例说明

本实例将演示如何使用简单控制器获取表单的数据。运行程序，在如图 19.1 所示的 index.jsp 页面中输入用户信息后单击“注册”按钮，将跳转到 reg.jsp 页面，显示用户输入的注册信息，如图 19.2 所示。

图 19.1　用户注册页

图 19.2　显示用户注册信息

关键技术

首先 Spring 的 DispatcherServlet 会将用户请求截获，然后转发给与请求名称对应的 Controller 控制器，在控制器中对数据进行处理后，再返回 reg.jsp 页面。

（1）编写 RegController.java 类文件，并让该类继承 AbstractController 简单控制器类。在该类中获取用户输入的表单信息，并将其保存在 Map 对象中，由视图模型对象返回。关键代码如下：

```
public class RegController extends AbstractController {
 protected ModelAndView handleRequestInternal(HttpServletRequest request,
                  HttpServletResponse response) throws Exception {
         return new ModelAndView("reg.jsp");                      //返回视图模型
 }
}
```

（2）编写 applicationContext.xml 配置文件，在该文件中配置上面编写的控制器。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
 xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">
 <!-- 配置控制器 Bean-->
 <bean name="/regController.html" class="com.jwy.controller.RegController"/>
</beans>
```

（3）在 web.xml 文件中配置核心控制器，并让其截获所有以.html 结尾的请求。关键代码如下：

```
<servlet>
<!-- 定义 Servlet 名称 -->
<servlet-name>dispatcherServlet</servlet-name>
<!-- Servlet 具体实现类 -->
<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
<!-- 初始化上下文对象 -->
<init-param>
        <!-- 参数名称 -->
        <param-name>contextConfigLocation</param-name>
        <!-- 加载配置文件 -->
```

```
        <param-value>/WEB-INF/applicationContext.xml</param-value>
 </init-param>
 <!-- 设置启动的优先级 -->
 <load-on-startup>1</load-on-startup>
</servlet>
<!-- 采用通配符映射所有.html 类型的请求 -->
<servlet-mapping>
 <servlet-name>dispatcherServlet</servlet-name>
 <url-pattern>*.html</url-pattern>
</servlet-mapping>
```

（4）编写首页文件 index.jsp，在该文件中定义 form 表单，让用户在表单中输入用户信息，并让该表单请求在 applicationContext.xml 文件中定义的 simpleController.html 控制器。关键代码如下：

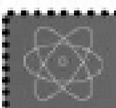
```
<form action="regController.html" method="post">
<table>
<tr>
        <td>输入用户名：</td>
        <td><input type="text" name="name"></td>
</tr>
<tr>
        <td>输入密码：</td>
        <td><input type="password" name="pwd"></td>
</tr>
<tr>
        <td>确认密码：</td>
        <td><input type="password" name="pwd1"></td>
</tr>
<tr>
        <td>电子邮箱：</td>
        <td><input type="text" name="mail"></td>
</tr>
<tr>
        <td colspan="2">
                <input type="submit" value="注册">
                <input type="reset" value="重置">
        </td>
</tr>
</table>
</form>
```

（5）编写 reg.jsp 页面，使用 EL 表达式将 Map 集合对象中的用户输入信息显示出来。关键代码如下：

```
<table align="center" border="1">
<tr>
        <td height="23"><span class="STYLE2">用户名：</span></td>
        <td height="23"><span class="STYLE2">${param.name }</span></td>
</tr>
<tr>
        <td height="23"><span class="STYLE2">密码：</span></td>
        <td height="23"><span class="STYLE2">${param.pwd }</span></td>
</tr>
<tr>
        <td height="23"><span class="STYLE2">邮箱：</span></td>
        <td height="23"><span class="STYLE2">${param.mail }</span></td>
</tr>
<tr>
        <td height="23" colspan="2" align="center"><a href="index.jsp" class="STYLE2">返回</a></td>
</tr>
</table>
```

心法领悟 497：Spring 的控制器。

通常情况下不会使用简单控制器作为控制器的父类，因为 Spring 在简单控制器的基础上提供了许多功能单一的更简单的控制器。可以使用这些现成的控制器降低开发控制器的负担，提高开发效率。



实例 498 参数映射控制器映射 JSP 页面

光盘位置：光盘\MR\19\498

高级

实用指数：★★★★

实例说明

WEB-INF 是受保护的目录，将 JSP 文件放在 WEB-INF 目录内，可以保证 JSP 文件不会被他人盗取。运行本实例，可以看到不同的进销存超链接，如图 19.3 所示。单击不同的超链接，即可进入对应的页面。



图 19.3　映射到不同 JSP 页面的超链接

关键技术

首先将 JSP 页面文件隐藏在 WEB-INF 保护目录下，然后使用视图解析器 InternalResourceViewResolver 通过给逻辑视图名添加前后缀的方式得到一个指向确定 JSP 页面的地址，最后应用 ParameterizableViewController 参数映射控制器控制不同页面的跳转。

设计过程

（1）在 applicationContext.xml 文件中配置参数映射控制器以及视图分解器。关键代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">
<!-- 定义视图分解器 -->
<bean id="viewResolver"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="viewClass">
                <value>org.springframework.web.servlet.view.InternalResourceView
                </value>
        </property>
        <!-- 设置前缀，即视图所在的路径 -->
        <property name="prefix" value="/WEB-INF/jsp/" />
        <!-- 设置后缀，即视图的扩展名 -->
        <property name="suffix" value=".jsp" />
</bean>
<bean name="/sys01.do" class="org.springframework.web.servlet.mvc.ParameterizableViewController">
        <property name="viewName" value="sys01"/>
</bean>
<bean name="/sys02.do" class="org.springframework.web.servlet.mvc.ParameterizableViewController">
        <property name="viewName" value="sys02"/>
</bean>
<bean name="/sys03.do" class="org.springframework.web.servlet.mvc.ParameterizableViewController">
        <property name="viewName" value="sys03"/>
</bean>
<bean name="/sys04.do" class="org.springframework.web.servlet.mvc.ParameterizableViewController">
        <property name="viewName" value="sys04"/>
</bean>
</beans>
```

（2）在 web.xml 文件中配置 Spring 控制器。关键代码如下：

```
<servlet>
<servlet-name>dispatcherServlet</servlet-name>
<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
<init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/applicationContext.xml</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
```

```
</servlet>
<servlet-mapping>
 <servlet-name>dispatcherServlet</servlet-name>
 <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

（3）编写 index.jsp 页面文件，在该文件中编写超链接。关键代码如下：

```
<center>
<img align="middle" src="images/main.jpg" width="800" height="660" border="0" usemap="#Map">
<map name="Map">
   <area shape="rect" coords="615,180,789,236" href="sys01.do">
<area shape="rect" coords="614,252,788,304" href="sys02.do">
<area shape="rect" coords="549,318,782,372" href="sys03.do">
<area shape="rect" coords="605,387,792,443" href="sys04.do">
</map>
</center>
```

（4）在 WEB-INF\JSP\文件夹下编写 sys01.jsp、sys02.jsp、sys03.jsp 和 sys04.jsp 页面文件。

秘笈心法

心法领悟 498：Spring 的视图解析器。

Spring MVC 控制器中的大部分方法都会返回一个 ModelAndView 类型的对象，在该对象中包含一个逻辑视图名，Spring 通过视图解析器将该逻辑视图名与实际的视图关联到一起。所有的视图解析器都实现了 ViewResolver 接口，该接口中只定义了一个方法 resolveViewName()，通过该方法可根据逻辑视图名和一个本地化对象得到一个视图对象。

通常使用 InternalResourceViewResolver 将逻辑视图名映射到保存在 WEB-INF 文件夹中的 JSP 页面文件。该方法通过“前缀+逻辑视图名+后缀”方式得到一个指向确定地址的 JSP 页面文件。

实例 499	文件名映射控制器映射 JSP 页面 光盘位置：光盘\MR\19\499	高级 实用指数：★★★

实例说明

虽然使用参数映射控制器可以为每个 JSP 页面的导向操作配置一个对应的控制器，但随着 JSP 页面文件数量的增多，Spring 配置的参数映射控制器也会随之增多。文件名映射控制器就是为了映射 URL 对 JSP 页面的控制，在 Spring 配置文件中只需要配置一个这样的控制器即可。运行本实例，可以看到使用文件名映射控制器映射到 JSP 页面的超链接，如图 19.4 所示，单击超链接即可进入对应的 JSP 页面。



图 19.4　文件名映射控制器映射 JSP 页面

关键技术

本实例中的文件名映射控制器主要用到了 UrlFilenameViewController 类。在 Spring 配置文件中只需配置一个文件名映射控制器即可。关键代码如下：

```
<!-- 文件名到视图的映射控制器 -->
<bean id="forword" class="org.springframework.web.servlet.mvc.UrlFilenameViewController"/>
<!-- 使用文件名映射控制器映射 JSP 页面 -->
<bean name="urlMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
<property name="mappings">
        <props>
                <prop key="/sys01.do">forword</prop>
                <prop key="/sys02.do">forword</prop>
```

```
                <prop key="/sys03.do">forword</prop>
            </props>
    </property>
</bean>
```

设计过程

（1）在 applicationContext.xml 文件中配置文件名映射控制器以及视图分解器。关键代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">
        <!-- 定义视图分解器 -->
<bean id="viewResolver"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="viewClass">
                <value>org.springframework.web.servlet.view.InternalResourceView
                </value>
        </property>
        <!-- 设置前缀，即视图所在的路径 -->
        <property name="prefix" value="/WEB-INF/jsp/" />
        <!-- 设置后缀，即视图的扩展名 -->
        <property name="suffix" value=".jsp" />
</bean>
<!-- 文件名到视图的映射控制器 -->
<bean id="forword" class="org.springframework.web.servlet.mvc.UrlFilenameViewController"/>
<!-- 映射 -->
<bean name="urlMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
        <property name="mappings">
                <props>
                        <prop key="/sys01.do">forword</prop>
                        <prop key="/sys02.do">forword</prop>
                        <prop key="/sys03.do">forword</prop>
                </props>
        </property>
</bean>
</beans>
```

（2）在 web.xml 文件中配置 Spring 控制器。关键代码如下：

```
<servlet>
        <servlet-name>dispatcherServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>/WEB-INF/applicationContext.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
        <servlet-name>dispatcherServlet</servlet-name>
        <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

（3）编写 index.jsp 页面文件，在该页面中编写超链接。关键代码如下：

```
<%@page contentType="text/html" pageEncoding="GBK"%>
<html>
<body>
        <a href="sys01.do">明日购物</a><br>
        <a href="sys02.do">鹏逊购物</a><br>
        <a href="sys03.do">G-shop</a><br>
<       /body>
</html>
```

（4）在 WEB-INF\JSP\文件夹下编写 sys01.jsp、sys02.jsp 和 sys03.jsp 页面文件。

秘笈心法

心法领悟 499：文件名映射控制器。

UrlFilenameViewController 文件名映射控制器也是用于请求转发的控制器。与参数映射控制器的区别在于，该控制器使用了“前缀+请求名+后缀”的方式来生成视图的名称。该类中声明了 prefix 与 suffix 属性，并且提供了 getter 与 setter 方法，所以不需要对该类进行扩展，即可直接使用。

实例 500　命令控制器获取 URL 中的参数查询信息

光盘位置：光盘\MR\19\500

高级

实用指数：★★★

实例说明

在 Web 项目中，常常需要根据用户请求中的参数进行操作，这时就用到了命令控制器。运行本实例，进入商品销售排行榜，如图 19.5 所示；单击商品名称，即可看到该商品的详细信息，如图 19.6 所示。



图 19.5　商品销售排行榜

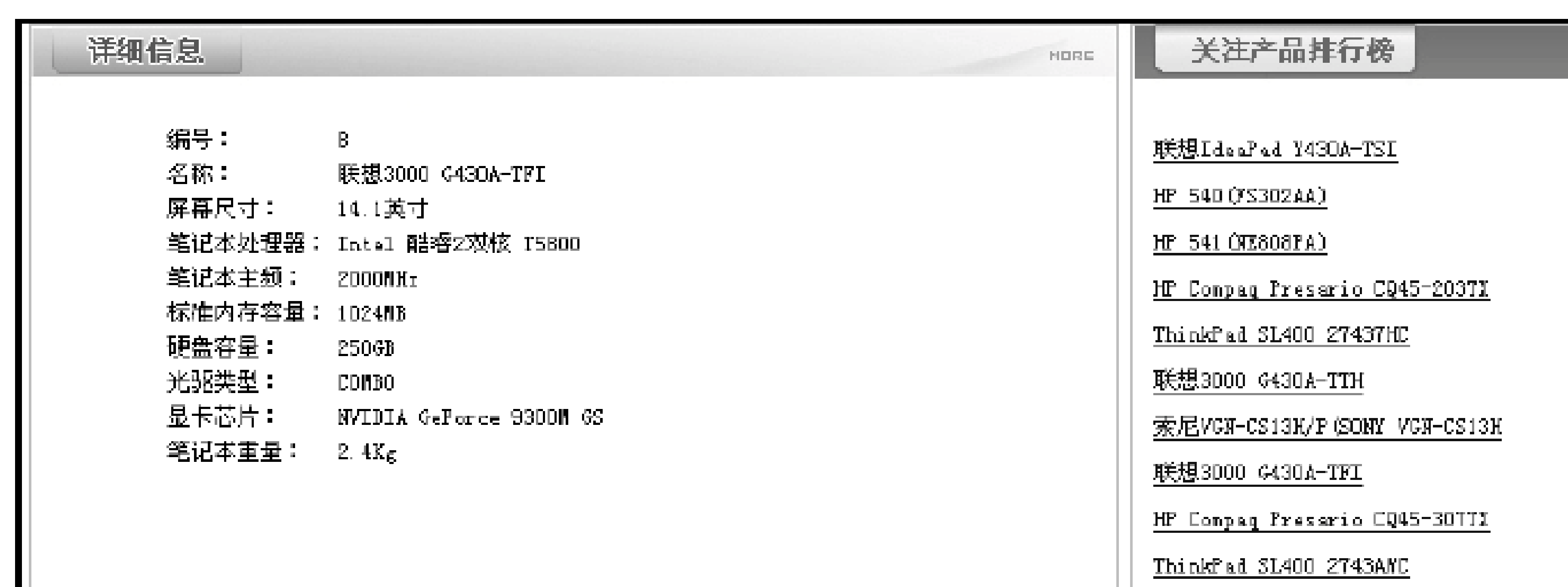


图 19.6　商品详细信息

本实例中的命令控制器要继承 AbstractCommandController，并在构造方法中使用 setCommandClass()方法设置命令对象的类型。关键代码如下：

```
public CommandController(){
 setCommandClass(Param.class);
}
```

然后在配置文件中配置该控制器。关键代码如下：

```
<bean name="/showInfo.do" class="com.jwy.controller.CommandController"/>
```

（1）编写 Param.java 类文件，用于封装表单中的参数。关键代码如下：

```
public class Param {
private Integer id;
public Integer getId() {
        return id;
}
public void setId(Integer id) {
        this.id = id;
}
}
```

（2）编写命令控制器 CommandController.java 文件，继承 AbstractCommandController 类。这样即可自动从请求中提取参数，并绑定到命令对象中。关键代码如下：

```
public class CommandController extends AbstractCommandController {
//设定命令对象的类型
public CommandController(){
        setCommandClass(Param.class);                          //设置封装参数的对象
}
protected ModelAndView handle(HttpServletRequest request,HttpServletResponse response, Object cmd, BindException arg3) throws Exception {
        List list = (List)request.getSession().getAttribute("list");   //获取 Session 中的 List 列表对象
        Param ps = (Param)cmd;                                 //获取封装参数的对象
        String[] str = (String[])list.get(ps.getId());         //根据参数找到对应的数据
        return new ModelAndView("showInfo","str",str);         //连同查询到的数据一同返回到视图
}
}
```

（3）在 applicationContext.xml 配置文件中配置命令控制器。关键代码如下：

```
<!-- 命令控制器 -->
<bean name="/showInfo.do" class="com.jwy.controller.CommandController"/>
```

（4）在 web.xml 文件中配置 dispatcherServlet。关键代码如下：

```
<servlet>
  <servlet-name>dispatcherServlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/applicationContext.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>dispatcherServlet</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

（5）编写 index.jsp 页面文件，在该页面文件中初始化一个保存有商品信息的 List 列表对象，并将其保存在会话对象中。关键代码如下：

```
<%
        List list = new ArrayList();
        String[] str0 = new String[10];
        str0[0] = "0";                                         //编号
        str0[1] = "联想 IdeaPad Y430A-TSI";
        str0[2] = "14.1 英寸";                                 //屏幕尺寸
        str0[3] = "Intel 酷睿 2 双核 T6400";                   //笔记本处理器
        str0[4] = "2000MHz";                                   //笔记本主频
        str0[5] = "2048MB";                                    //标准内存容量
        str0[6] = "250GB";                                     //硬盘容量
        str0[7] = "DVD 刻录机";                                //光驱类型
        str0[8] = "NVIDIA GeForce 9300M GS";                   //显卡芯片
        str0[9] = "2.35Kg";                                    //笔记本重量
        list.add(str0);
        ……                                                     //省略部分代码
        session.setAttribute("list",list);
%>
<a href="listInfo.jsp">关注产品排行榜</a>
```

（6）编写 listInfo.jsp 文件，显示出排行榜内容。关键代码如下：

```
<--关注产品排行榜-->
<%
 List list = (List)session.getAttribute("list");
 for(int i=0;i<list.size();i++){
        String[] str = (String[])list.get(i);
%>
 <a href="showInfo.do?id=<%= str[0] %>"><%= str[1] %></a><br>
<%
 }
%>
```

（7）编写 showInfo.jsp 文件，显示商品的详细信息。关键代码如下：

```
<body>
商品详细信息<br>
        编号：${str[0] }<br>
        名称：${str[1] }<br>
        屏幕尺寸：${str[2] }<br>
        笔记本处理器：${str[3] }<br>
        笔记本主频：${str[4] }<br>
        标准内存容量：${str[5] }<br>
        硬盘容量：${str[6] }<br>
        光驱类型：${str[7] }<br>
        显卡芯片：${str[8] }<br>
        笔记本重量：${str[9] }<br>
</body>
```

秘笈心法

心法领悟 500：多个控制器映射。

在 Web 应用程序中使用一个控制器映射很难完成解析所有请求的工作，这时就要配置多个控制器映射，并通过 order 属性指定各映射的优先级。dispatcherServlet 会顺序使用控制器映射解析 URL 中的请求，直到返回正确的结果。

实例 501　利用表单控制器向图书信息表中添加数据

光盘位置：光盘\MR\19\501

高级　实用指数：★★★★

实例说明

通过 Spring 的 MVC 框架开发 Web 程序时，经常要将表单中的数据存储到后台数据库中。Spring 框架提供了功能强大的表单控制器 SimpleFormController，开发者可以使用这个控制器获取表单中的信息。运行本实例，在如图 19.7 所示的表单中输入图书信息，然后单击“保存”按钮，即可将图书信息保存到数据库中。

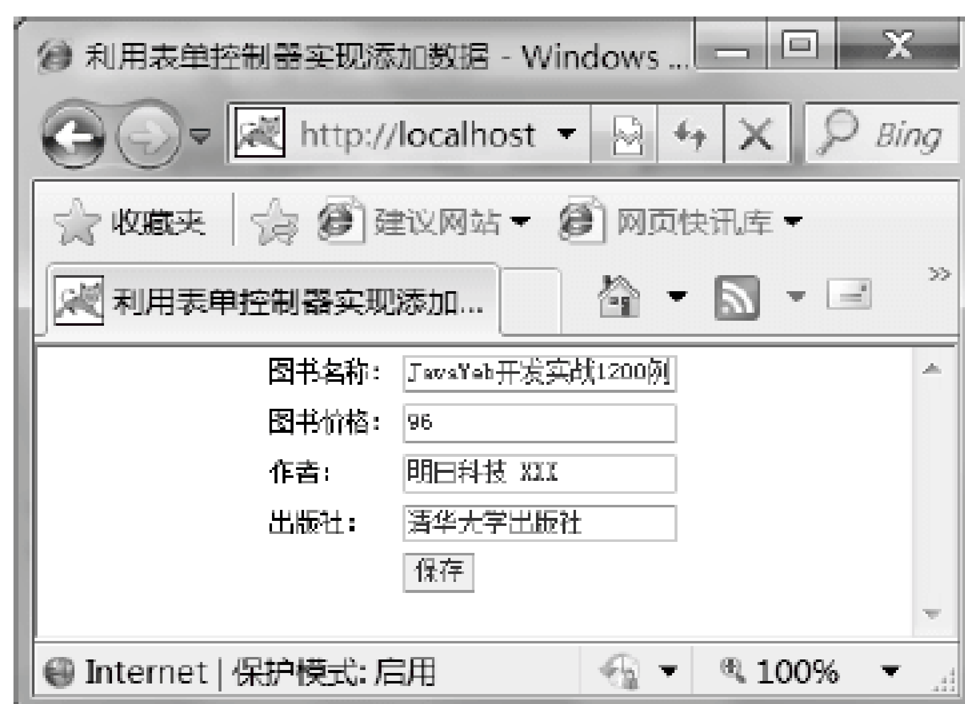


图 19.7　利用表单控制器实现添加图书信息

关键技术

首先需要自定义一个控制器，实现表单控制器 SimpleFormController；然后在控制器所提供的 onSubmit()方法中通过绑定对象参数 object 来完成表单的绑定操作，绑定成功之后，生成相应的 SQL 语句；最后调用 Dao 对象的 execute()方法完成数据添加操作。

（1）创建 BookInfo.java 类文件，作用是完成对图书信息表单的绑定，并且在类中定义相对应的私有变量

以及相应的 get()或 set()方法。

（2）创建 BookDao.java 类文件，在该类中引入 JDBC 模板类；然后定义一个类型为 JdbcTemplate 的对象 jtl，该对象使用 Spring 的 IoC 依赖注入特征；再定义 executeSql()方法，用来执行相应的 SQL 语句。其关键代码如下：

```
import org.springframework.jdbc.core.JdbcTemplate;
public class BookDao{
private JdbcTemplate jtl = null;
    public JdbcTemplate getJtl() {
        return jtl;
    }
    public void setJtl(JdbcTemplate jtl) {
          this.jtl = jtl;
    }
    public void executeSql(String insertSql){
          jtl.execute(insertSql);
    }
}
```

（3）创建 BookController.java 类文件，继承 SimpleFormController 类，并定义 BookDao 的实例对象 bookDao，该对象使用 Spring 的 IoC 依赖注入。接下来，在 onSubmit()方法中获取绑定表单的图书对象，然后保存到数据库。其关键代码如下：

```
public class BookController extends SimpleFormController {
 private BookDao bookDao;
 public BookDao getBookDao() {
        return bookDao;
 }
 public void setBookDao(BookDao bookDao) {
        this.bookDao = bookDao;
 }
 protected Object fromBackingObject(HttpServletRequest request)throws Exception{
        request.setCharacterEncoding("GBK");
        return super.formBackingObject(request);
 }
 protected ModelAndView onSubmit(HttpServletRequest request,HttpServletResponse response,Object obj,BindException bind){
        BookInfo book = (BookInfo)obj;//获取图书信息对象
        try {
                String name=book.getBookName();
                float price = book.getPrice();
                String author =book.getAuthor();
                String bookmaker = book.getBookmaker();
                String sql = "insert into tb_book(bookName,price,author,bookmaker) values('"
                        +name+"','"+price+"','"+author+"','"+bookmaker+"')";
                bookDao.executeSql(sql);//保存到数据库
        } catch (Exception e) {
                e.printStackTrace();
        }
        Map<String,String> message = new HashMap<String, String>();
        message.put("msg", "保存成功！ ");
        return new ModelAndView("index",message);//返回到相应的视图
 }
}
```

（4）在 WEB-INF 目录下创建配置文件 bean-config.xml，用来完成 Spring 的 MVC 配置操作。其关键代码如下：

```
<beans>
 <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource">
                <ref local="dataSource"/>
        </property>
 </bean>
 <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource" >
        <property name="driverClassName">
                <value>com.mysql.jdbc.Driver</value>
        </property>
        <property name="url">
                <value>jdbc:mysql://localhost:3306/db_database19</value>
        </property>
        <property name="username">
```

```
                <value>root</value>
        </property>
        <property name="password">
                <value>111</value>
        </property>
</bean>
<bean id="daosupport" class="com.lh.dao.BookDao">
        <property name="jtl">
                <ref bean="jdbcTemplate"/>
        </property>
</bean>
<bean id="viewResolver"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="viewClass">
                <value>org.springframework.web.servlet.view.JstlView</value>
        </property>
        <property name="prefix">
                <value>/</value>
        </property>
        <property name="suffix">
                <value>.jsp</value>
        </property>
</bean>
<bean name="/save.do" class="com.lh.controller.BookController">
        <property name="commandClass">
                <value>com.lh.entity.BookInfo</value>
        </property>
        <property name="bookDao">
                <ref bean="daosupport"/>
        </property>
</bean>
</beans>
```

（5）创建 index.jsp 页面文件，该页面用于输入信息，并且设置表单的 action 为 save.do，这样在提交时会根据 Spring 的配置将表单数据绑定到表单控制器中。其关键代码如下：

```
<form method="post" action="save.do" name="myform">
     <table align="center">
        <tr>
                <td>图书名称：</td>
                <td>
                <input type="text" name="bookName"/>
                </td>
        </tr>
        <tr>
                <td>图书价格：</td>
                <td>
                <input type="text" name="price"/>
                </td>
        </tr>
        <tr>
                <td>作者：</td>
                <td>
                <input type="text" name="author" />
                </td>
        </tr>
        <tr>
                <td>出版社：</td>
                <td>
                <input type="text" name="bookmaker"/>
                </td>
        </tr>
        <tr>
                <td></td>
                <td>
                <input type="submit" value="保存"/>
                </td>
        </tr>
        <tr>
                <td></td>
```

```
            <td>
            ${msg}
            </td>
        </tr>
    </table>
</form>
```

秘笈心法

心法领悟 501：绑定表单数据。

应用 Spring MVC 提供的表单控制器获取表单非常方便，只要把页面中表单元素的名称与 Bean 中的属性名称设置为相同，表单控制器就会将表单中的数据封装成一个 Bean 对象。

实例 502 利用表单控制器验证用户登录

光盘位置：光盘\MR\19\502

高级

实用指数：★★★

实例说明

表单在网页中有着非常重要的作用，主要负责与用户交互时采集数据。通常在实现用户登录时，需要对用户名和密码进行校验，只有输入正确才允许登录。运行本实例，首先在浏览器地址栏中输入 http://localhost:8080/502/userLogin.html，即可看到用户登录页面，如图 19.8 所示。输入用户名 mr 与密码 mrsoft，单击“登录”按钮，即可进入欢迎页面，如图 19.9 所示。

图 19.8　用户登录页面

图 19.9　欢迎页面

关键技术

Spring MVC 的表单控制器与正常的表单处理流程有所不同，该控制器从被请求时起就已经开始控制表单了。表单的载入、提交、转向结果页都是由表单控制器来控制。

（1）编写实体类 User 封装表单中的用户名与密码，使用 setter 与 getter 方法读出数据。关键代码如下：

```
public class User {
 private String userName;
 private String userPwd;
 public String getUserName() {
         return userName;
 }
 public void setUserName(String userName) {
         this.userName = userName;
 }
 ……//省略部分代码
}
```

（2）编写控制器类 UserLoginController，并让该类继承 SimpleFormController 表单控制器类，然后重写 onSubmit()方法，在该方法中对用户提交的用户名与密码进行验证，如果验证成功则进入到欢迎页面，如果验证失败则返回到输入页面并显示错误信息。关键代码如下：

```
public class UserLoginController extends SimpleFormController {
 protected ModelAndView onSubmit(Object command) throws Exception {
         User user = (User)command;
         String usereName = user.getUserName();
```

```
        String userPwd = user.getUserPwd();
        Map map = new HashMap();
        if("mr".equals(usereName)&&"mrsoft".equals(userPwd)){
                map.put("user", user);
                return new ModelAndView(getSuccessView(),"map",map);
        }else{
                map.put("error", "用户名或密码不正确，请重新输入！ ");
                return new ModelAndView(getFormView(),"map",map);
        }
}
}
```

（3）在上面的代码中，onSubmit()方法传入一个 Object 类型的对象，该对象就是封装了表单数据的实体对象。该对象是从何而来的呢？下面就来看一下 Spring 的配置文件 applicationContext.xml，该配置文件关键代码如下：

```
<!-- 表单控制器 -->
<bean name="/userLogin.html" class="com.jwy.controller.UserLoginController">
        <property name="commandClass">
                <value>com.jwy.controller.User</value>
        </property>
        <!-- 输入表单数据页面 -->
        <property name="formView">
                <value>index.jsp</value>
        </property>
        <!-- 表单提交后转入页面 -->
        <property name="successView">
                <value>login.jsp</value>
        </property>
</bean>
```

（4）编写 index.jsp 页面文件。该页面中含有一个<form>元素，但需要注意的是，此处并没有为该元素设置 action 属性。关键代码如下：

```
    <form method="post">
        <center>${map.error }</center>
    <table align="center">
        <tr>
                <td height="23"><span class="STYLE3">输入用户名：</span></td>
                <td height="23"><input name="userName" type="text"></td>
        </tr>
        <tr>
                <td height="23"><span class="STYLE3">输入密码：</span></td>
                <td height="23"><input name="userPwd" type="password"></td>
        </tr>
        <tr>
                <td height="23" colspan="2" align="center">
                        <input type="submit" value="登录">
                        <input type="reset" value="重置">
                </td>
        </tr>
    </table>
    </form>
```

（5）编写 login.jsp 页面文件，登录验证成功后会转向该页面，并在其中显示出欢迎信息。关键代码如下：

```
<center>
系统登录成功<br>${map.user.userName}，欢迎光临！
</center>
```

（6）配置 web.xml 文件。web.xml 文件与前面实例的配置一样，也是配置 DispatcherServlet 让其截获所有以.html 结尾的请求。

心法领悟 502：表单控制器的配置。

在 Spring 的配置文件中，首先配置了一个表单控制器 UserLoginController，然后将 commandClass 参数设置为 com.jwy.controller.User 类型，这样在表单被提交时 Spring 会自动将表单中的数据封装到 User 类型的对象，并传递到 onSubmit()方法中。接下来，还配置了 formView 参数与 successView 参数，这两个参数分别用来输入信息的表单页面与表单处理完成之后的转向页面。



实例 503 利用多动作控制器跳转到不同页面

光盘位置：光盘\MR\19\503

高级
实用指数：★★★

实例说明

一个动作对应一个控制器的做法具有很大的局限性，因为很多时候要完成的工作都具有一定的相似性，这时就要在不同的控制器中编写大量完成相同功能的代码。对此 Spring MVC 提供了一个可以完成多个动作的 MultiActionController 控制器。本实例将使用多动作控制器访问不同的页面，如图 19.10 所示。



图 19.10　一个控制器进入不同页面

关键技术

在 Spring 的 MVC 中提供了一个 MultiActionController 多动作控制器。与其他的控制器完全不同，MultiActionController 可以在一个控制器中定义多个方法。只要继承 MultiActionController 类即可实现多动作控制器。在该控制器中定义的方法的返回值可以是 ModelAndView、Map 或 void，并且有两个参数，分别为 HttpServletRequest 与 HttpServletResponse。

设计过程

（1）编写多动作控制器类 SampleMultiActionController，继承 MultiActionController 类，并在该类中编写 vipLogin()方法进入注册页面、userLogin()方法进入登录页面。关键代码如下：

```
public class SampleMultiActionController extends MultiActionController {
 //进入注册页面
 public ModelAndView vipLogin(HttpServletRequest req, HttpServletResponse res)
                throws ServletRequestBindingException, IOException {
        return new ModelAndView("vipLogin");
 }
 //进入登录页面
 public ModelAndView userLogin(HttpServletRequest req,HttpServletResponse res) throws ServletRequestBindingException,
                IOException {
        return new ModelAndView("userLogin");
 }
}
```

（2）编写 applicationContext.xml 配置文件。首先映射 URL，在 WEB-INF 文件夹内创建 applicationContext.xml 配置文件。接下来，配置方法名解析器。关键代码如下：

```
<!-- URL 映射 -->
<bean id="urlMapping"
 class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
 <property name="mappings">
        <props>
                <!-- sample.do 请求映射到 sampleMultiActionController 类 -->
                <prop key="sample.do">sampleMultiActionController</prop>
        </props>
 </property>
</bean>
<!-- 方法名解析器 -->
<bean id="paraMethodResolver"
 class="org.springframework.web.servlet.mvc.multiaction.ParameterMethodNameResolver">
 <!-- 根据参数指定方法名 -->
 <property name="paramName" value="whichMethod" />
</bean>
```

```
<!-- 为 sampleMultiActionController 类设置方法名解析器 -->
<bean id="sampleMultiActionController" class="com.jwy.SampleMultiActionController">
 <property name="methodNameResolver">
        <ref bean="paraMethodResolver" />
 </property>
</bean>
```

（3）编写 index.jsp 页面文件，该页面中有两个超链接请求同一个控制器，但请求的参数不同，参数直接对应控制器中的方法名称。关键代码如下：

```
<%@page contentType="text/html" pageEncoding="GBK"%>
<html>
<head>
<title>多动作控制器</title>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
</head>
<body>
<center><img src="images/bj.jpg" border="0" usemap="#Map">
<map name="Map"><area shape="rect" coords="292,381,409,408" href="sample.do?whichMethod=vipLogin">
<area shape="rect" coords="418,380,538,410" href="sample.do?whichMethod=userLogin">
</map></center>
</body>
</html>
```

（4）编写 vipLogin.jsp 与 userLogin.jsp 用户登录页面。

秘笈心法

心法领悟 503：多动作控制器的使用场合。

顾名思义，MultiActionController 多动作控制器就是一个控制器可以执行很多动作。在应用程序的开发过程中，如果程序非常庞大，往往需要用户定义很多控制器，这样非常不利于管理。这时可以考虑将功能相近的一类控制器放在一起，例如对用户的查询、增加、修改、删除。

实例 504 利用向导控制器实现用户注册

光盘位置：光盘\MR\19\504

高级

实用指数：★★★

实例说明

在用户注册时，为了不让用户感到需要填写的信息太多，常常会把需要用户填写的信息分成多个页面，分别让用户填写。这时可以使用 Spring 中的向导控制器来完成这项工作。运行本实例，单击“进入用户注册页面”超链接，进入用户注册的“第一步：登录信息”页面，如图 19.11 所示。完成相关信息的填写后，单击“下一步”按钮，进入“第二步：详细资料”页面，如图 19.12 所示。完成详细资料的填写后，单击“下一步”按钮，进入“第三步：联系方式”页面，如图 19.13 所示。完成所有信息的填写后，单击“确定”按钮，进入“注册信息”页面，在其中显示出用户填写的所有信息，如图 19.14 所示。

图 19.11　用户注册第一步

图 19.12　用户注册第二步

图 19.13　用户注册第三步

图 19.14　显示用户输入的注册信息

关键技术

在配置文件中可以使用<bean>标签的 pages 属性配置向导页面的顺序，也可以通过<list><value></value></list>的方式对向导页面顺序进行配置，还可以使用逗号分隔字符串的方式进行配置。本实例中使用的就是逗号分隔符的方式。关键代码如下：

```
<property name="pages" value="onePage,twoPage,threePage" />
```

（1）编写 User.java 实体类文件，用于封装用户输入的表单信息。关键代码如下：

```
public class User {
    private String userName;        //用户名
    private String pwd;             //密码
    private String pwd1;            //确认密码
    private String qq;              //QQ 号码
    private String mail;            //电子邮箱
    private String tel;             //电话
    private String addr;            //地址
    private String name;            //姓名
    private String age;             //年龄
    private String sex;             //性别
    private String high;            //身高
    private String weight;          //体重
    public String getPwd1() {
        return pwd1;
    }
    public void setPwd1(String pwd1) {
        this.pwd1 = pwd1;
    }
    ……//省略部分代码
}
```

（2）编写向导控制器类 GuideController.java，该类继承自 AbstractWizardFormController 类。关键代码如下：

```
public class GuideController extends AbstractWizardFormController {
 private String cancelView; //取消时跳转的页面
 private String finishView;  //完成后跳转的页面
 public void setCancelView(String cancelView) {
        this.cancelView = cancelView;
 }
```

```
public void setFinishView(String finishView) {
        this.finishView = finishView;
}
//最后提交表单时执行的方法
protected ModelAndView processFinish(HttpServletRequest arg0,HttpServletResponse arg1, Object arg2, BindException arg3)
                throws Exception {
        User fullUser = (User)arg2;
        return new ModelAndView(finishView,"user",fullUser);
}
//取消时执行的方法
protected ModelAndView processCancel(HttpServletRequest request,HttpServletResponse response, Object command, BindException errors)
                throws Exception {
        return new ModelAndView(cancelView);
}
}
```

（3）在 applicationContext.xml 文件中配置表单对象与向导页面中的视图。关键代码如下：

```
<bean name="user" class="com.jwy.User" />
<bean name="/userReg.do" class="com.jwy.GuideController">
<!-- 封装表单的对象 -->
<property name="commandClass" value="com.jwy.User" />
<!-- 向导页面 -->
<property name="pages" value="onePage,twoPage,threePage" />
<!-- 取消后转向的视图 -->
<property name="cancelView" value="index" />
<!-- 向导完成后转向的视图 -->
<property name="finishView" value="ok" />
</bean>
```

（4）在 web.xml 文件中配置 dispatcherServlet。关键代码如下：

```
<servlet>
<servlet-name>dispatcherServlet</servlet-name>
<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
<init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/applicationContext.xml</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>dispatcherServlet</servlet-name>
<url-pattern>*.do</url-pattern>
</servlet-mapping>
```

（5）编写 onePage.jsp、twoPage.jsp、threePage.jsp 页面文件，并使用_targetX（X 为要转到的页面的索引）的命名方式为“上一步”与“下一步”按钮命名。例如，要转到 onePage.jsp 页面就是_target0。关键代码如下：

```
<form:form>
姓名：<form:input path="name"/><br>
年龄：<form:input path="age"/><br>
性别：<form:input path="sex"/><br>
身高：<form:input path="high"/><br>
体重：<form:input path="weight"/><br>
<input type="submit" name="_target0" value="上一步">
<input type="submit" name="_target2" value="下一步"><br>
</form:form>
```

另外还有“确定”按钮，该按钮的 name 属性为_finish，与控制器中的 processFinish()方法相对应。本实例中“确定”按钮的代码如下：

```
<input type="submit" name="_finish" value="确定" >
```

心法领悟 504：AbstractWizardFormController 解析。

实际上，AbstractWizardFormController 实现类本质上依然是像 SimpleFormController 那样，分两个阶段来管理表单页面的处理，只不过是从逻辑上将单个表单页面划分为多个表单页面，而最终绑定数据的 Command 对象

却只有一个。AbstractWizardFormController 将根据_targetX 参数决定显示表单页面的哪一部分，对应到视图就是显示哪个向导页面。

实例 505 利用多动作控制器操作员工信息表的数据

光盘位置：光盘\MR\19\505

高级

实用指数：★★★

实例说明

本实例将通过 Spring 的 MVC 框架提供的 MultiActionController 控制器，实现员工信息的查询与删除。运行程序，在如图 19.15 所示页面中单击“查询员工信息”超链接，将显示出所有的员工信息；单击员工信息列表中的“删除”按钮，将删除指定的员工信息。



图 19.15　员工信息的查询与删除

关键技术

当使用 MultiActionController 控制器时，必须与 org.springframework.web.servlet.mvc.multiaction. PropertiesMethodNameResolver 类同时使用，PropertiesMethodNameResolver 类用来配置控制器中的各个方法要被执行。配置方法如下：

```
<bean id="paraMethodResolver"
      class="org.springframework.web.servlet.mvc.multiaction.PropertiesMethodNameResolver">
 <property name="mappings">
      <props>
      <prop key="/queryemp.do">QueryEmp</prop>
      <prop key="/deleteemp.do">DeleteEmp</prop>
      </props>
</property>
</bean>
```

通过 Spring 的 JDBC 模板类 JdbcTemplate 完成数据查询操作，语法如下：

```
JdbcTemplate.queryForList(String selectsql);
```

参数说明

selectsql：查询语句的字符串。

（1）创建 EmpDao.java 类文件。首先导入需要的程序类包；然后定义 JdbcTemplate 模板的实例对象 jtl，该对象使用 Spring 的 IoC 依赖注入特征；在这个类中定义两个方法 executeSql()和 querySql()，分别用来完成对数据库的删除和查询操作。关键代码如下：

```
public class EmpDao {
 private JdbcTemplate jtl = null;
```

```
public JdbcTemplate getJtl() {
        return jtl;
}
public void setJtl(JdbcTemplate jtl) {
        this.jtl = jtl;
}
public void executeSql(String deleteSql){
        jtl.execute(deleteSql);
}
public List querySql(String selectsql){
        return jtl.queryForList(selectsql);
}
}
```

（2）创建 EmpAction.java 类文件，该类继承了 MultiActionController 类。在该类中定义一个类型为 EmpDao 的对象 empDao 用于执行数据库操作，该对象使用 Spring 的 IoC 依赖注入；再定义一个查询方法 QueryEmp() 用来查询满足条件的数据。关键代码如下：

```
public class EmpAction extends MultiActionController {
private EmpDao empDao;
public EmpDao getEmpDao() {
        return empDao;
}
public void setEmpDao(EmpDao empDao) {
        this.empDao = empDao;
}
public ModelAndView QueryEmp(HttpServletRequest request, HttpServletResponse res) {
        String sqlSelect = "select * from tb_employeeinfo ";
        List empList = empDao.querySql(sqlSelect);
        Map map = new HashMap();
        map.put("empList", empList);
        return new ModelAndView("index",map);
}
}
```

（3）在 WEB-INF 文件夹下创建 bean_config.xml 配置文件，用来对控制器的请求进行操作。关键代码如下：

```
<bean id="daosupport" class="com.lh.dao.EmpDao">                              //配置 EmpDao
        <property name="jtl">
                <ref bean="jdbcTemplate"/>
        </property>
</bean>
<bean id="viewResolver"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">     //配置视图解析器
        <property name="viewClass">
                <value>org.springframework.web.servlet.view.JstlView</value>
        </property>
        <property name="prefix">
                <value>/</value>
        </property>
        <property name="suffix">
                <value>.jsp</value>
        </property>
</bean>
<bean id="paraMethodResolver"
        class="org.springframework.web.servlet.mvc.multiaction.PropertiesMethodNameResolver">
        <property name="mappings">
                <props>
                        <prop key="/queryemp.do">QueryEmp</prop>
                        <prop key="/deletemp.do">DeleteEmp</prop>
                </props>
        </property>
</bean>
<bean name="/*emp.do" class="com.lh.action.EmpAction">
        <property name="methodNameResolver">
                <ref bean="paraMethodResolver"/>
        </property>
        <property name="empDao">
                <ref local="daosupport"/>
        </property>
</bean>
```

（4）创建 index.jsp 页面文件，用来对控制器的请求进行操作。关键代码如下：

```
<c:forEach var="emp" items="${empList}">
        <tr>
            <td height="28" align="center" class="style4">
                <div align="center">
                    <c:out value="${emp.name}" />
                </div>
            </td>
                <td height="28" align="center" class="style4">
                <div align="center">
                    <c:out value="${emp.sex}"/>
                </div>
            </td>
            <td height="28" align="center" class="style4">
                <div align="center">
                    <c:out value="${emp.age}"/>
                </div>
            </td>
            <td height="28" align="center" class="style4">
                <div align="center">
                    <c:out value="${emp.tel}"/>

                </div>
            </td>
             <td height="28" align="center" class="style4">
                <div align="center">
                    <c:out value="${emp.addr}" />
                </div>
            </td>
            <td height="28" align="center" class="style4">
                <div align="center">
                    <input type="button" value="删除" onclick="window.location.href('deletemp.do?id=${emp.id}')" />
                </div>
            </td>
        </tr>
    </c:forEach>
```

秘笈心法

心法领悟 505：MultiActionController。

MultiActionController 继承了 AbstractController，所以也就拥有了 AbstractController 所处理的那些通用关注点的能力。

19.2 在线通讯录

下面使用 Spring MVC 框架实现一个简单的在线通讯录，用户可以向通讯录内添加联系人，可以修改联系人的联系方式，还可以删除已经存在的联系人。

实例 506	添加新联系人 光盘位置：光盘\MR\19\506	高级 实用指数：★★★

本实例将实现在线通讯录的添加新联系人功能。运行程序，将显示出所有的通讯录信息。单击“添加新记录”超链接，将进入输入新记录页面，如图 19.16 所示。在表单中输入员工的信息，然后单击“确定”按钮，即可添加新联系人。



图 19.16　添加新联系人

关键技术

在线通讯录包含了增、删、改、查等基本功能，本实例只实现添加新联系人功能。在此通过 Spring MVC 框架实现。通过对多动作控制器的了解，可以知道应用它来控制增、删、改、查等功能比较合适，所以此处应用多动作控制器来实现。

（1）编写 AddrBook.java 实体类文件，用于封装通讯录的信息。代码如下：

```
public class AddrBook {
 private Integer id;                         //编号
 private String name;                        //姓名
 private String company;                     //公司
 private String job;                         //职位
 private String tel;                         //办公电话
 private String mobile;                      //移动电话
 private String mail;                        //电子邮件
 private String fax;                         //传真
 public Integer getId() {
        return id;
 }
 public void setId(Integer id) {
        this.id = id;
 }
 ……//省略部分 getter 与 setter 方法
}
```

（2）编写 IAddrBookDao.java 接口，在该接口中声明对数据表操作的方法。关键代码如下：

```
public interface IAddrBookDao {
 public void insert(AddrBook addrBook);      //向数据表插入数据
 public void update(AddrBook addrBook);      //更新数据表中数据
 public void delete(Integer id);             //按主键 id 删除数据表中数据
 public List<Map> findByAll();               //查询数据表中所有数据
 public AddrBook findById(Integer id);       //按主键 id 查询数据
}
```

（3）编写 IAddrBookDao 接口的实现类 AddrBookDao，该类继承自 JdbcDaoSupport 类，实现 IAddrBookDao 接口中的 insert()方法，用于添加新联系人。关键代码如下：

```
 public void insert(AddrBook addrBook) {
        Object[] o = { addrBook.getName(), addrBook.getCompany(),
                    addrBook.getJob(), addrBook.getTel(),
                    addrBook.getMobile(), addrBook.getFax(), addrBook.getMail() };
        getJdbcTemplate().update("INSERT INTO "+ "tb_addrBook(name,company,job,tel,mobile,fax,mail) "
                          + "values (?,?,?,?,?,?,?)",o);
 }
```

（4）编写多动作控制器类 AddrBookController，该类继承自 MultiActionController 类，在该类中声明

IAddrBookDao 类型的私有成员变量 addrBookDao，通过 setter 方法为其赋值。关键代码如下：

```
public class AddrBookController extends MultiActionController {
 private IAddrBookDao addrBookDao;
 public void setAddrBookDao(IAddrBookDao addrBookDao) {
        this.addrBookDao = addrBookDao;
 }
}
```

在该类中添加 insertAndUpdate()方法，在该方法中首先从页面中获取表单中的数据信息，然后通过用户编号 id 属性判断执行插入还是更新方法。如果 id 属性的值为 0，说明是一条新的记录，这时调用插入数据的方法，否则执行更新数据记录的方法。

```
public ModelAndView insertAndUpdate(HttpServletRequest request,HttpServletResponse response){
        AddrBook addrBook = new AddrBook();
        addrBook.setName(request.getParameter("name"));
        addrBook.setCompany(request.getParameter("company"));
        addrBook.setJob(request.getParameter("job"));
        addrBook.setTel(request.getParameter("tel"));
        addrBook.setMobile(request.getParameter("mobile"));
        addrBook.setMail(request.getParameter("mail"));
        addrBook.setFax(request.getParameter("fax"));
        addrBook.setId(Integer.valueOf(request.getParameter("id")));
        if(addrBook.getId()==0){
                addrBookDao.insert(addrBook);           //执行插入方法
        }else{
                addrBookDao.update(addrBook);           //执行更新方法
        }
        return findByAll(request, response);
}
```

（5）配置数据库操作类，为其注入数据源，配置多动作解析器和之前编写的多动作控制器。关键代码如下：

```
<!-- 配置 AddrBookDao 类 -->
<bean id="addrBookDao" class="com.jwy.dao.AddrBookDao">
 <!-- 注入数据源 -->
 <property name="dataSource" ref="dataSource" />
</bean>

<!-- 配置 AddrBookController 类 -->
<bean name="/addrBook.html" class="com.jwy.controller.AddrBookController">
 <!-- 注入 addrBookDao -->
 <property name="addrBookDao" ref="addrBookDao" />
 <!-- 注入多动作解析器 -->
 <property name="methodNameResolver">
        <ref bean="paraMethodResolver" />
 </property>
</bean>
<!-- 配置多动作解析器 -->
<bean id="paraMethodResolver"
 class="org.springframework.web.servlet.mvc.multiaction.ParameterMethodNameResolver">
 <property name="paramName" value="method" />
</bean>
```

（6）编写 insertAndUpdate.jsp 页面文件，在该页面中通过一个隐藏表单来判断添加与修改数据信息。关键代码如下：

```
<form name="f1" method="post" action="addrBook.html?method=insertAndUpdate">
 <table align="center">
        <tr>
                <td height="24">姓名：</td>
                <td height="24">
                        <input type="text" name="name" value="${addrBook.name==null?":addrBook.name}">
                        <input type="hidden" name="id" value="${addrBook.id==null?0:addrBook.id}">
           </td>
        </tr>
        <tr>
                <td height="24">工作单位：</td>
                <td height="24">
                        <input type="text" name="company"value="${addrBook.company==null?":addrBook.company}"></td>
        </tr>
        <tr>
```

```
                <td height="24">职位：</td>
                <td height="24">
                    <input type="text" name="job" value="${addrBook.job==null?":addrBook.job}"></td>
            </tr>
            <tr>
                <td height="24">办公电话：</td>
                <td height="24">
                    <input type="text" name="tel" value="${addrBook.tel==null?":addrBook.tel}"></td>
            </tr>
            <tr>
                <td height="24">移动电话：</td>
                <td height="24">
                    <input type="text" name="mobile" value="${addrBook.mobile==null?":addrBook.mobile}"></td>
            </tr>
            <tr>
                <td height="24">传真：</td>
                <td height="24">
                    <input type="text" name="fax" value="${addrBook.fax==null?":addrBook.fax}"></td>
            </tr>
            <tr>
                <td height="24">电子邮箱：</td>
                <td height="24">
                    <input type="text" name="mail" value="${addrBook.mail==null?":addrBook.mail}"></td>
            </tr>
            <tr>
                <td height="24" colspan="2">
                    <div align="center">
                      <input type="submit" value="确定">  <input type="reset"   value="重置">
                    </div></td>
            </tr>
        </table>
</form>
```

秘笈心法

心法领悟 506：MultiActionController 的助理 MethodNameResoler。

MethodNameResoler 的主要作用是帮助 MultiActionController 决定当前 Web 请求应该交给哪个方法处理。在 Spring MVC 框架内默认提供了如下 3 种策略实现：

- InternalPathMethodNameResolver
- PropertiesMethodNameResolver
- ParameterMethodNameResolver

实例 507　修改联系人信息

光盘位置：光盘\MR\19\507

高级

实用指数：★★★

实例说明

本实例将实现在线通讯录的修改联系人信息的功能。运行本程序，首先会显示出所有的通讯录信息列表，如图 19.17 所示。单击列表中某个员工信息后的“修改”超链接，将进入到修改联系人信息页面，如图 19.18 所示。在此页面中修改员工信息，然后单击“确定”按钮，即可完成修改。

图 19.17　通讯录信息列表

图 19.18　修改联系人信息

关键技术

实现修改当前的联系人信息，首先需要读取出当前的联系人信息并添加到表单中；然后获取当前联系人信息的 id 保存到表单的隐藏域内，因为需要根据该 id 进行更新；最后将表单提交给 Spring 的控制器，调用相应的方法更新数据库。

（1）在 AddrBookDao 类中，实现 IAddrBookDao 接口中的 update()方法，用于更新联系人。关键代码如下：

```
public void update(AddrBook addrBook) {
        Object[] o = { addrBook.getName(), addrBook.getCompany(),
                        addrBook.getJob(), addrBook.getTel(), addrBook.getMobile(),
                        addrBook.getFax(), addrBook.getMail(),addrBook.getId()};
        getJdbcTemplate().update("UPDATE tb_addrBook SET name=?,company=?,job=?,tel=?,mobile=?,fax=?,mail=? WHERE id=?",o);
}
```

（2）在该类中添加 insertAndUpdate()方法，在该方法中首先从页面中获取表单中的数据信息，然后通过用户编号 id 属性判断执行插入还是更新方法。如果 id 属性的值为 0，说明是一条新的记录，这时调用插入数据的方法，否则执行更新数据记录的方法。

```
public ModelAndView insertAndUpdate(HttpServletRequest request,HttpServletResponse response){
        AddrBook addrBook = new AddrBook();
        addrBook.setName(request.getParameter("name"));
        addrBook.setCompany(request.getParameter("company"));
        addrBook.setJob(request.getParameter("job"));
        addrBook.setTel(request.getParameter("tel"));
        addrBook.setMobile(request.getParameter("mobile"));
        addrBook.setMail(request.getParameter("mail"));
        addrBook.setFax(request.getParameter("fax"));
        addrBook.setId(Integer.valueOf(request.getParameter("id")));
        if(addrBook.getId()==0){
                addrBookDao.insert(addrBook);           //执行插入方法
        }else{
                addrBookDao.update(addrBook);           //执行更新方法
        }
        return findByAll(request, response);
}
```

（3）编写 insertAndUpdate.jsp 页面文件，在该页面中通过一个隐藏表单来判断添加与修改数据信息。关键代码如下：

```
<form name="f1" method="post" action="addrBook.html?method=insertAndUpdate">
<table align="center">
        <tr>
                <td height="24">姓名：</td>
                <td height="24">
                        <input type="text" name="name" value="${addrBook.name==null?'':addrBook.name}">
                        <input type="hidden" name="id" value="${addrBook.id==null?0:addrBook.id}">
           </td>
        </tr>
        <tr>
                <td height="24">工作单位：</td>
                <td height="24">
                        <input type="text" name="company"value="${addrBook.company==null?'':addrBook.company}"></td>
        </tr>
        <tr>
                <td height="24">职位：</td>
```

```
            <td height="24">
                <input type="text" name="job" value="${addrBook.job==null?'':addrBook.job}"></td>
        </tr>
        <tr>
            <td height="24">办公电话：</td>
            <td height="24">
                <input type="text" name="tel" value="${addrBook.tel==null?'':addrBook.tel}"></td>
        </tr>
        <tr>
            <td height="24">移动电话：</td>
            <td height="24">
                <input type="text" name="mobile" value="${addrBook.mobile==null?'':addrBook.mobile}"></td>
        </tr>
        <tr>
            <td height="24">传真：</td>
            <td height="24">
                <input type="text" name="fax" value="${addrBook.fax==null?'':addrBook.fax}"></td>
        </tr>
        <tr>
            <td height="24">电子邮箱：</td>
            <td height="24">
                <input type="text" name="mail" value="${addrBook.mail==null?'':addrBook.mail}"></td>
        </tr>
        <tr>
            <td height="24" colspan="2">
                <div align="center">
                  <input type="submit" value="确定">  <input type="reset"   value="重置">
                </div></td>
        </tr>
    </table>
</form>
```

秘笈心法

心法领悟 507：InternalPathMethodNameResolver。

如果没有为 MultiActionController 明确指定任何 MethodNameResolver，那么 InternalPathMethodNameResolver 将作为默认的 MethodNameResolver 实现，以进行 Web 请求与具体处理方法间直接的映射解析。

实例 508	删除联系人 光盘位置：光盘\MR\19\508	高级 实用指数：★★★

实例说明

本实例将实现在线通讯录的删除联系人信息的功能。运行本程序，首先会显示出所有的通讯录信息列表，如图 19.19 所示。单击列表中某个员工信息后的“删除”超链接，即可将其删除。

图 19.19　删除联系人

关键技术

在删除通讯录中联系人时，同样需要根据 id 进行。因此，在通讯录信息列表中单击“删除”超链接时，会

将当前的通讯信息的 id 作为请求参数传递到 Spring 的控制器中，在控制器中的相关方法中再调用数据库的删除方法删除当前的联系人。

设计过程

（1）在通讯录信息列表中，添加“删除”超链接，并将通讯信息的 id 作为参数进行传递。代码如下：

```
<a href="addrBook.html?method=delete&id=${item.id}">删除</a>
```

（2）在 AddrBookDao 类中，实现 IAddrBookDao 接口中的 delete()方法，用于删除联系人。关键代码如下：

```
public void delete(Integer id) {
        getJdbcTemplate().update("DELETE FROM tb_addrBook WHERE id="+id);
}
```

（3）在多动作控制器的实现类 AddrBookController 中编写 delete()方法，在该方法中获取当前要删除通讯信息的 id，然后调用 Dao 类的 delete()方法删除指定的联系人。代码如下：

```
public ModelAndView delete(HttpServletRequest request,HttpServletResponse response){
        Integer id = Integer.valueOf(request.getParameter("id"));
        addrBookDao.delete(id);
        return findByAll(request, response);
}
```

秘笈心法

心法领悟 508：PropertiesMethodNameResolver。

PropertiesMethodNameResolver 与 InternalPathMethodNameResolver 的唯一相同点在于，它们都是基于请求的 URL 进行映射，二者有共同的父类 AbstractUrlMethodNameResolver。相比 InternalPathMethodNameResolver，PropertiesMethodNameResolver 更灵活。

实例 509 查询通讯录中的信息

光盘位置：光盘\MR\19\509

高级

实用指数：★★★

实例说明

本实例将实现在线通讯录的查询功能。运行本程序，会直接显示从数据库中查询出的所有的通讯录信息列表，如图 19.20 所示。

员工通讯录

添加新记录

姓名	工作单位	职位	办公电话	移动电话	传真	电子邮件	操作
张三	明日科技	java程序员	89089***	1594306****	89089***	jing*****@163.com	修改 删除
张四	明日科技	.net程序员	0431-84972266	1594306****	8779****	111@mr.com	修改 删除
王二*	明日	程序员	8765****	1276689****	0431-8876****	tt**@163.com	修改 删除

图 19.20 查询通讯录信息

在默认情况下，在浏览器中输入当前项目的 URL 地址，就会直接显示出所有的通讯录信息。这是由于在 web.xml 文件中配置了一个默认访问路径，当在浏览器中直接访问 Web 项目的 URL 时同样会显示出所有的通讯录信息。

在 web.xml 文件中，首先通过<welcome-file-list>配置默认访问页面为 index.jsp；然后在 index.jsp 中通过 JS 将路径跳转到查询页面中，并传递一个 method 参数 findByAll，这样 Spring 的 MultiActionController 控制器就会根据 ParameterMethodNameResolver 的配置将请求的参数值作为映射的方法名，而且在 MultiActionController 实

现类中也定义了一个 findByAll()方法，因此会直接调用 findByAll()方法查询出所有的数据再转向相应的视图。

设计过程

（1）在 index.jsp 页中，通过 JavaScript 设置请求的路径并传递查询数据的参数。关键代码如下：

```
<body>
  <script type="text/javascript">
    window.location = "addrBook.html?method=findByAll"
  </script>
</body>
```

（2）在 AddrBookDao 类中，实现 IAddrBookDao 接口中的 findByAll()方法，用于查询所有的联系人。关键代码如下：

```
public List<Map> findByAll() {
        List list = getJdbcTemplate().queryForList("SELECT * FROM tb_addrBook");
        return list;
}
```

（3）在多动作控制器的实现类 AddrBookController 中编写 findByAll()方法，在该方法中调用 Dao 类的 findByAll()方法查询所有的通讯录并返回一个 List，然后添加到视图中。代码如下：

```
public ModelAndView findByAll(HttpServletRequest request,HttpServletResponse response){
        List<Map> list = addrBookDao.findByAll();
        return new ModelAndView("show","list",list);
}
```

（4）在 show.jsp 页中，应用 JSTL 的 forEach 标签遍历 List 集合，读取出所有的通讯录信息。代码如下：

```
<c:forEach items="${list}" var="item" varStatus="i">
                <tr>
                        <td height="24" align="center" class="td2">${item.name}</td>
                        <td height="24" align="center" class="td2">${item.company}</td>
                        <td height="24" align="center" class="td2">${item.job}</td>
                        <td height="24" align="center" class="td2">${item.tel}</td>
                        <td height="24" align="center" class="td2">${item.mobile}</td>
                        <td height="24" align="center" class="td2">${item.fax}</td>
                        <td height="24" align="center" class="td2">${item.mail}</td>
                        <td height="24" align="center" class="td2"><a href="addrBook.html?method=findById&id=${item.id}">修改</a></td>
                        <td height="24" align="center" class="td2"><a href="addrBook.html?method=delete&id=${item.id}">删除</a></td>
                </tr>
</c:forEach>
```

秘笈心法

心法领悟 509：ParameterMethodNameResolver。

ParameterMethodNameResolver 允许用户根据请求中的某个参数的值作为映射的方法名，也允许使用请求中的一组参数映射处理方法名称。ParameterMethodNameResolver 默认检测的参数名称为 action，可以在配置时修改这个默认的参数名称。代码如下：

```
<bean id="paraMethodResolver"
        class="org.springframework.web.servlet.mvc.multiaction.ParameterMethodNameResolver">
        <property name="paramName" value="method" />
</bean>
```

19.3　图书信息管理

实例 510	添加图书信息 光盘位置：光盘\MR\19\510	高级 实用指数：★★★

实例说明

本实例是通过 Spring 的 MVC 技术实现图书信息管理的图书添加功能。运行程序，在显示的图书信息列表

中单击“添加图书”超链接，将跳转到图书信息添加页面，如图 19.21 所示。输入图书信息后单击“添加”按钮，图书信息即可添加到数据库中。

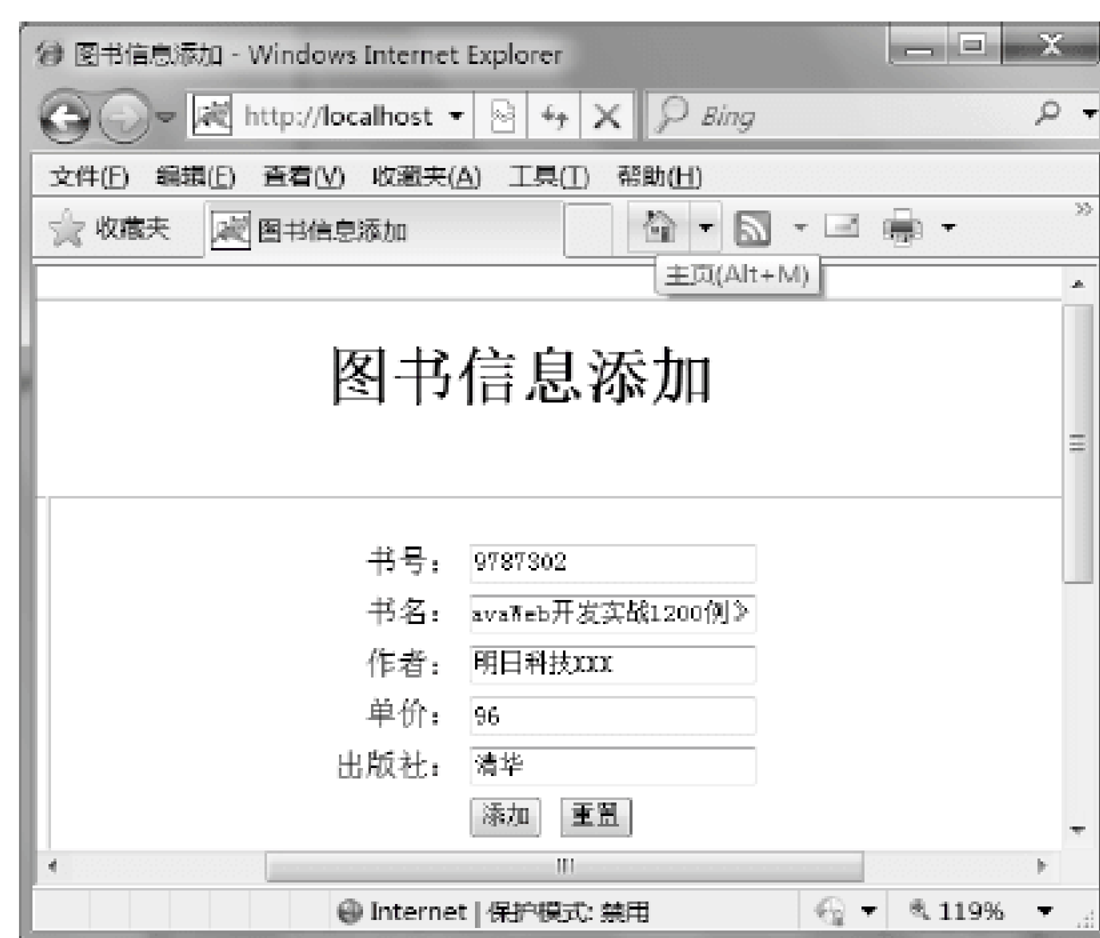


图 19.21　添加图书信息

关键技术

在实现添加图书信息时，主要应用 Spring MVC 中的表单控制器来实现。此处需要创建一个 SimpleFormController 控制器的实现类，并实现其 onSubmit()方法，在 onSubmit()方法中获取绑定表单的数据，并保存到数据库中。

（1）创建一个用于封装图书表单的 FormBean 类，需要通过 Spring 将表单与这个 Bean 进行绑定。关键代码如下：

```
public class FormBean {
	private String ISBN;
	private String name;
	private String writer;
	private String price;
	private String publisher;
	……//省略的 getter 和 setter 方法
}
```

（2）创建 SimpleFormController 表单控制器的实现类 BookAddController，在 onSubmit()方法中获取绑定表单的数据，然后进行保存。代码如下：

```
public class BookAddController extends SimpleFormController {
private Dao dao;
public void setDao(Dao dao) {
	this.dao = dao;
}
@Override
protected ModelAndView onSubmit(Object command) throws Exception {
	FormBean bean = (FormBean) command;
	if (bean.getISBN() != null && bean.getName() != null
			&& !bean.getISBN().isEmpty() && !bean.getName().isEmpty()) {
		dao.addBook(bean);
	}
	List list = dao.loadBooks();
	return new ModelAndView(getSuccessView(), "list", list);
}
}
```

（3）创建 Dao 类，编写添加图书信息的方法和查询图书信息的方法。代码如下：

```
public void addBook(FormBean bean) {
	getJdbcTemplate().execute(
			"insert into tb_books values(null,'" + bean.getISBN() + "','"
					+ bean.getName() + "','" + bean.getWriter() + "','"
					+ bean.getPrice() + "','" + bean.getPublisher() + "')");
}
public List loadBooks() {
	List list = getJdbcTemplate().queryForList("select * from tb_books");
	return list;
}
```

（4）在 Spring 的配置文件中配置表单控制器和 Bean 的依赖注入关系。关键代码如下：

```
<bean id="addBook" class="com.lzw.BookAddController">
	<property name="commandClass">
		<value>com.lzw.FormBean</value>
	</property>
	<property name="formView">
		<value>addBook</value>
	</property>
	<property name="successView">
```

```
            <value>index</value>
        </property>
        <property name="dao" ref="dao" />
</bean>
```

秘笈心法

心法领悟 510：设置密码。

在实际应用中，一般不会在数据库中以明文的方式保存用户密码，因为这样很容易造成密码泄露。可以将密码加密后以密文的方式进行保存。另外一种更有效的方法是仅保存密码的 MD5 摘要，因为相同的两个字符串 MD5 摘要值相同，因此可以通过比较摘要值是否相等来判断用户输入的密码是否正确。

实例 511　修改图书信息

光盘位置：光盘\MR\19\511

高级

实用指数：★★★

实例说明

本实例通过 Spring 的 MVC 实现图书信息管理中的修改图书信息功能。运行程序，在图书信息列表中某一记录的后面单击“修改”超链接，将跳转到图书信息修改页面，如图 19.22 所示。输入修改信息后单击“修改”按钮，即可完成修改并将修改后的图书信息保存到数据库。

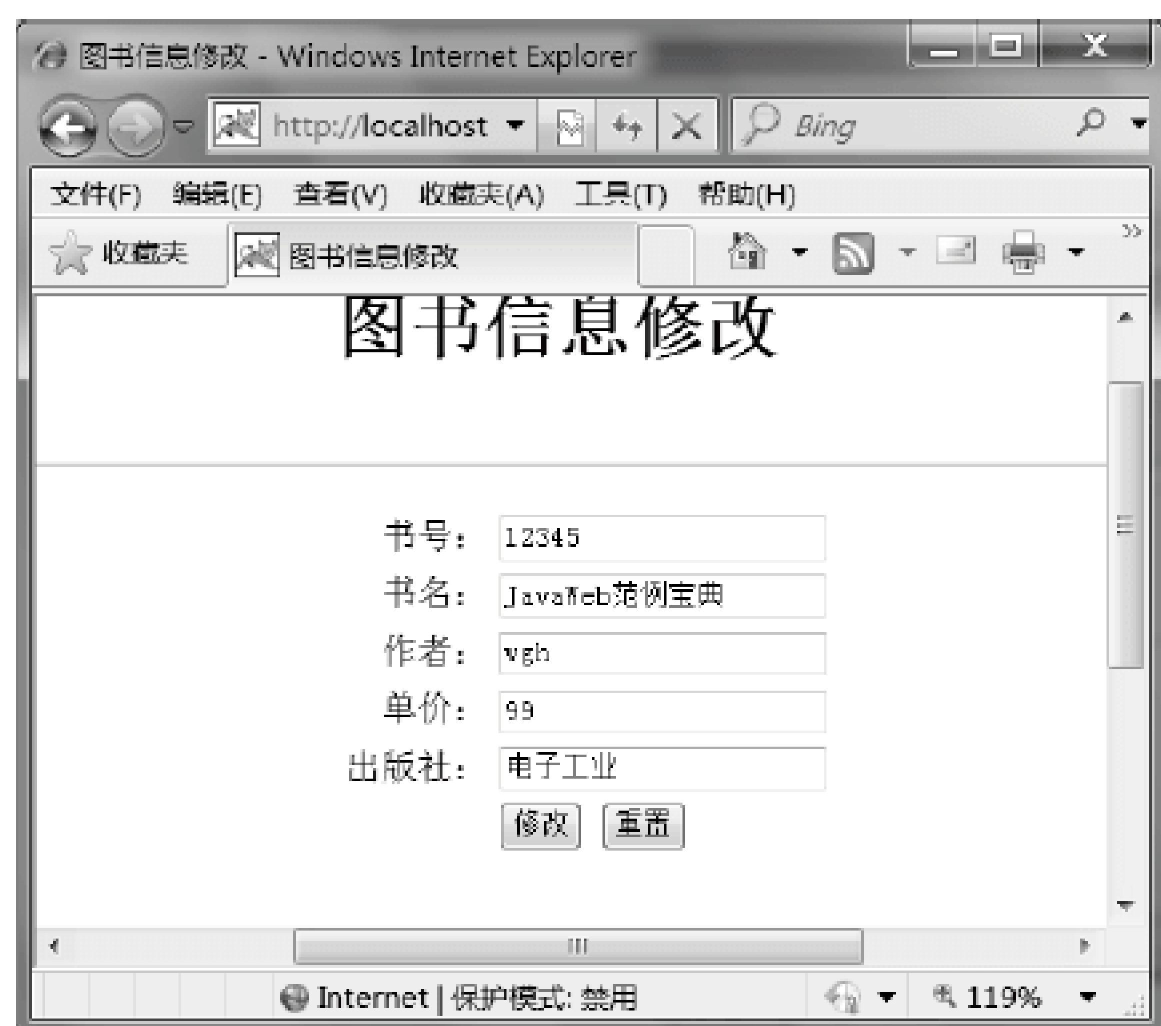


图 19.22　图书信息修改

关键技术

实现修改图书信息，主要使用 Spring 提供的表单控制器。首先需要在控制器中获取当前要修改的这一条图书信息的唯一 id，根据此 id 查询数据库，获取当前图书信息并显示在表单中，在表单中修改图书信息并提交后，在表单控制器的 onSubmit()方法中将数据更新到数据库中。

（1）创建 UpdateBookController 类实现表单控制器，实现更新图书信息。代码如下：

```
public class UpdateBookController extends SimpleFormController {
 private Dao dao;
 public void setDao(Dao dao) {
        this.dao = dao;
 }
 @Override
 protected ModelAndView showForm(HttpServletRequest request,
                HttpServletResponse response, BindException errors)
                throws Exception {
        String id = request.getParameter("id");
        Map book = dao.getBook(id);
        return new ModelAndView(getFormView(),book);
 }
 @Override
 protected ModelAndView onSubmit(HttpServletRequest request,
                HttpServletResponse response, Object command, BindException errors)
                throws Exception {
        FormBean bean=(FormBean) command;
        dao.updateBook(bean);
        request.getRequestDispatcher("index.html").forward(request, response);
        return null;
 }
}
```

（2）在 Dao 类中编写更新图书信息的方法和根据图书 id 查询图书信息的方法。代码如下：

```
public void updateBook(FormBean bean) {
        getJdbcTemplate().execute(
                        "update tb_books set " +
                        "isbn='" + bean.getISBN() + "',name='"
                        + bean.getName() + "',writer='" + bean.getWriter() + "',price='"
                        + bean.getPrice() + "',publisher='" + bean.getPublisher() + "'");
}
public Map getBook(String id) {
        return getJdbcTemplate().queryForMap(
                        "select * from tb_books where id='" + id + "'");
}
```

（3）在 Spring 的配置文件中配置更新图书信息的表单控制器和依赖注入关系。关键代码如下：

```
<bean id="updateBookController" class="com.lzw.UpdateBookController">
        <property name="dao" ref="dao" />
        <property name="commandClass">
                <value>com.lzw.FormBean</value>
        </property>
        <property name="formView">
                <value>updateBook</value>
        </property>
        <property name="successView" value="index" />
</bean>
```

秘笈心法

心法领悟 511：Bean 的作用域。

Spring 2.0 增加了若干个新的 Bean 作用域，在 Web 应用环境下，可以使用 request、session 和 globalSession 的 Bean 作用域；此外，还允许通过编程的方式定义新的 Bean 作用域。

实例 512　删除图书信息

光盘位置：光盘\MR\19\512

高级

实用指数：★★★

实例说明

本实例通过 Spring 的 MVC 实现图书信息管理中的删除图书信息功能。运行程序，在图书信息列表中的某一条数据后单击“删除”超链接，即可删除当前图书信息，如图 19.23 所示。

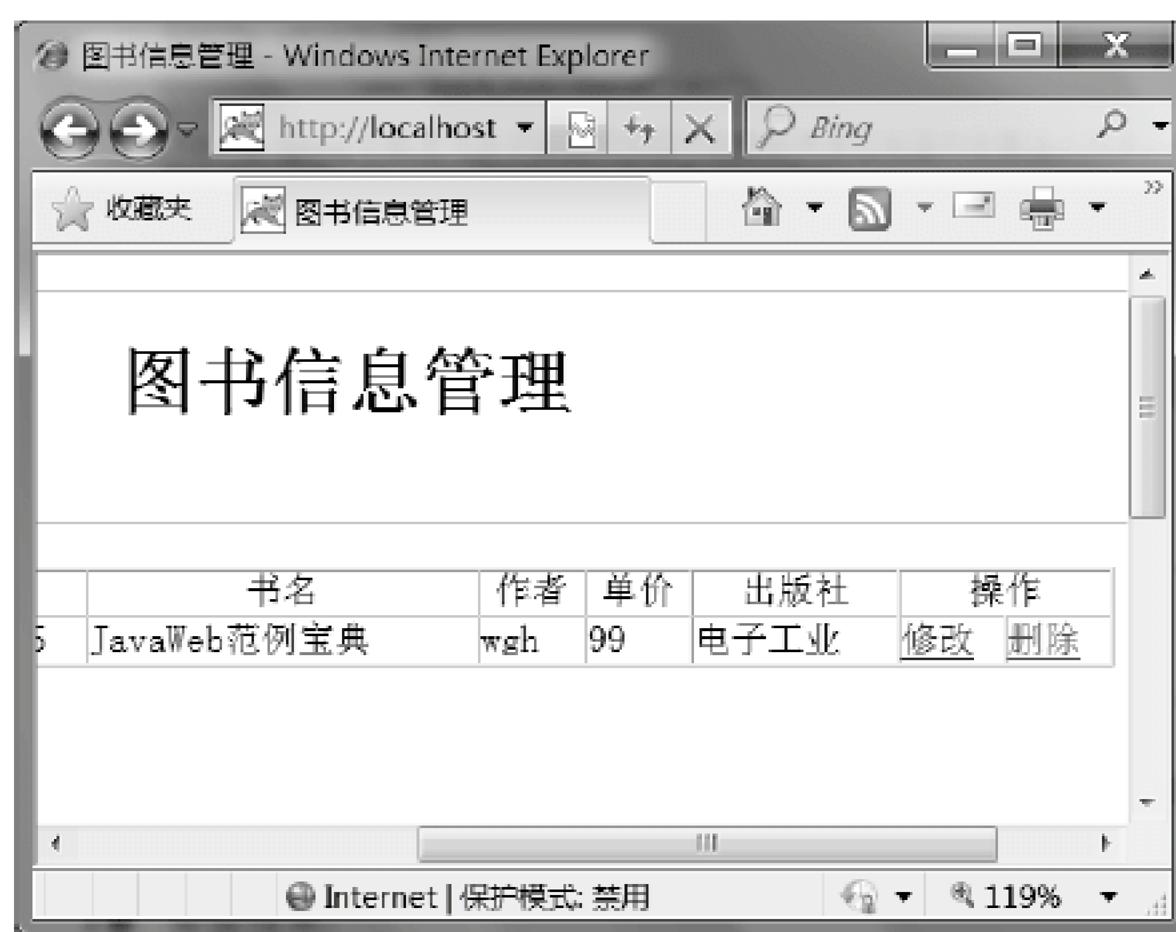


图 19.23　删除图书信息

关键技术

实现删除图书信息时，主要用到 AbstractController 控制器。在该控制器的实现类中重写 handleRequestInternal()方法，在此方法中获取当前要删除的图书 id，然后调用 Dao 类的 delete()方法执行删除操作。

（1）创建删除图书信息的控制器 DelBookController。代码如下：

```
public class DelBookController extends AbstractController {
 private Dao dao;

 public void setDao(Dao dao) {
        this.dao = dao;
 }

 @Override
```

```
protected ModelAndView handleRequestInternal(HttpServletRequest req,
            HttpServletResponse res) throws Exception {
        String id = req.getParameter("id");
        dao.delete(id);
        req.getRequestDispatcher("index.html").forward(req, res);
        return null;
    }
}
```

（2）在 Dao 类中编写删除图书信息的方法。关键代码如下：

```
public void delete(String id) {
        getJdbcTemplate().update("delete from tb_books where id='" + id + "'");
}
```

（3）在 Spring 的配置文件中配置控制器和 Bean 的依赖注入关系，关键代码如下：

```
<bean id="delBookController" class="com.lzw.DelBookController">
        <property name="dao" ref="dao" />
</bean>
```

秘笈心法

心法领悟 512：在 Eclipse 中查询 Spring 源码类所在的包。

Spring 源码类所在的包层次结构清晰，但是包名很长，从而给 Spring 的学习带来了一些不便。在 Eclipse 中，可以通过 Ctrl+Shift+R 组合键输入类名，Eclipse 将自动查找出这个类。

实例 513	查询图书信息 光盘位置：光盘\MR\19\513	高级 实用指数：★★★

实例说明

本实例通过 Spring 的 MVC 实现图书信息管理中的查询图书信息功能。运行程序，将显示查询到的图书信息列表，如图 19.24 所示。

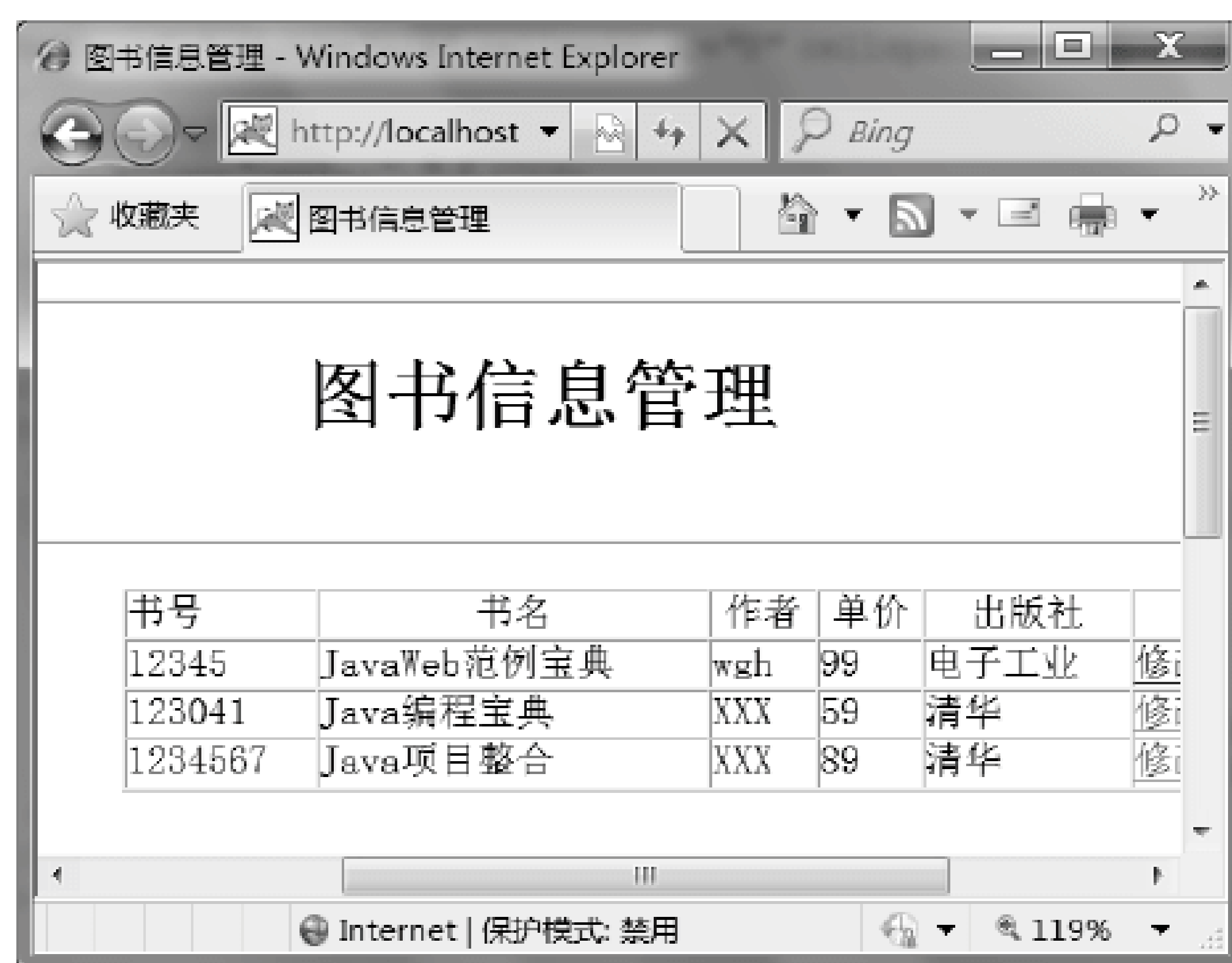


图 19.24　图书信息列表

关键技术

通过 Spring 的 MVC 实现查询图书信息，此处应用的是 AbstractController 控制器，在其 handleRequestInternal()方法中调用 Dao 类的查询方法查询图书信息。

（1）创建 BookListController 控制器，实现查询图书信息。代码如下：

```
public class BookListController extends AbstractController {

    private Dao dao;
```

```
public void setDao(Dao dao) {
        this.dao = dao;
}

@Override
protected ModelAndView handleRequestInternal(HttpServletRequest arg0,
                HttpServletResponse arg1) throws Exception {
        // TODO Auto-generated method stub
        List list = dao.loadBooks();
        return new ModelAndView("index", "list", list);
}
}
```

（2）在 Spring 的配置文件中配置查询图书的控制器和相关 Bean 的依赖注入关系。代码如下：

```
<bean id="bookList" class="com.lzw.BookListController">
        <property name="dao" ref="dao" />
</bean>
```

（3）在 Spring 的配置文件中配置视图解析器和文件名映射控制器，当在浏览器中输入项目 URL 路径访问时，会通过文件名控制器的配置请求不同的控制器，经过处理之后会通过视图解析器的配置跳转到不同页面。关键代码如下：

```
<!-- 配置视图解析器 -->
<bean id="viewResolver"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="viewClass">
                <value>org.springframework.web.servlet.view.JstlView
                </value>
        </property>
        <property name="prefix">
                <value>/WEB-INF/jsp/</value>
        </property>
        <property name="suffix">
                <value>.jsp</value>
        </property>
</bean>
<!-- 文件名映射控制器 -->
<bean id="urlMapping"
        class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
        <property name="mappings">
                <props>
                        <prop key="/index.html">bookList</prop>
                        <prop key="/addBook.html">addBook</prop>
                        <prop key="/updateBook.html">updateBookController</prop>
                        <prop key="/delBook.html">delBookController</prop>
                </props>
        </property>
</bean>
```

心法领悟 513：在 Eclipse 中实现大小写转换。

在程序开发中，经常可以遇到一句代码中既有大写又有小写的情况。例如，在 SQL 语句中，关键字和函数通常会大写，其他的表名、字段名等内容则为小写。这样反复切换很麻烦。此时可以将整句代码统一大小写。一般的 IDE 都提供了大小写转换的快捷键，如在 Eclipse 中，用户可以通过 Ctrl+Shift+Y 或 Ctrl+Shift+X 组合键对选择的代码进行大小写转换。

第 6 篇

网站安全与架构模式篇

⏭ 第 20 章　网站性能优化与安全策略

⏭ 第 21 章　设计模式与架构

第20章

网站性能优化与安全策略

- 文件保护
- 漏洞防护与数据加密
- 获取客户端信息

20.1　文 件 保 护

实例 514　防止用户直接输入地址访问 JSP 文件

光盘位置：光盘\MR\20\514

中级

实用指数：★★☆

实例说明

网站可以由多个动态页面组成，并且每个动态页面之间都存在着联系。为了保证网站内信息资源的安全，程序员应禁止浏览者不通过登录页面而强行进入其他页面进行浏览。运行本实例，当浏览者没有通过登录页面，而是直接通过“不登录直接访问欢迎页面”超链接进入欢迎页面时，将会弹出如图 20.1 所示的提示信息。

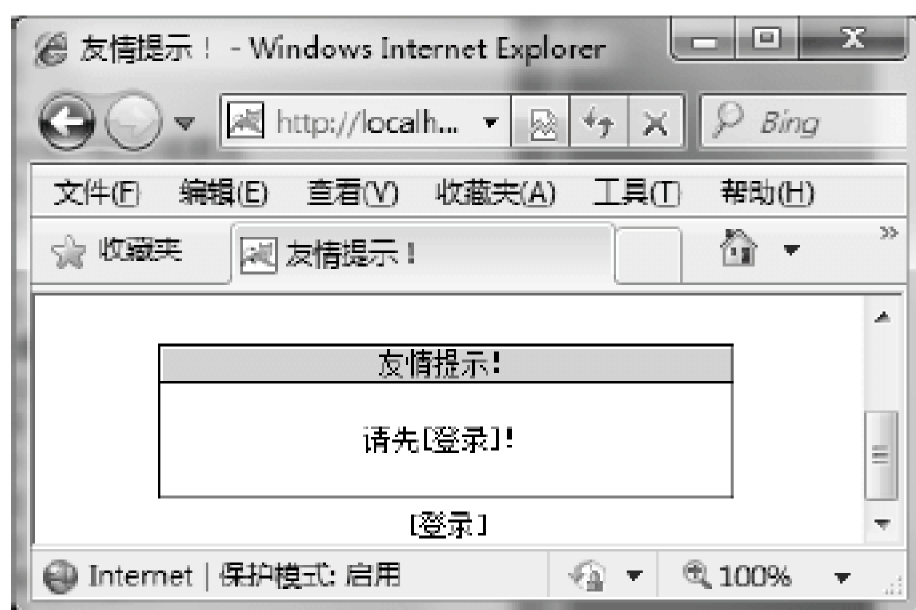


图 20.1　防止用户直接输入地址访问 JSP 文件

关键技术

当用户登录成功后，将有关用户登录的信息，如用户名或存储用户信息的对象存入 Session 对象的属性中。在其他每个页面中首先要对该属性进行判断，看其是否存储了用户信息。如果没有，则强制转到登录页面要求用户进行登录。

（1）创建 DoyouLogon 类验证并存储用户信息。关键代码如下：

```
package com.safe.UserLogon;
public class DoyouLogon {
    private String username="";                         //用户输入的用户名
    private String userpassword="";                     //用户输入的密码
    private Checkstr check=new Checkstr();
    public DoyouLogon(){}
    ……//省略了属性的 getXXX()和 setXXX()方法
    public String checkuser(){
        String backstr="";
        if(this.username.equals("")){
            backstr+="<li>请输入<b>用户名！ </b></li><br>";
        }
        if(this.userpassword.equals("")){
            backstr+="<li>请输入<b>密 码！ </b></li>";
        }
        return backstr;
    }
}
```

（2）创建登录的首页 index.jsp。关键代码如下：

```
<%@ page contentType="text/html;charset=GBK"%>
<%   session.invalidate();   %>
     <form action="dologon.jsp">
        <table>
           <tr bgcolor="lightgrey">
              <td align="center">请先登录</td>
```

```
                </tr>
                <tr height="50">
                    <td align="center">
                        用户名：<input type="text" name="username" size="30">
                        <br>
                        密  码：<input type="password" name="userpassword" size="30" redisplay="false">
                    </td>
                </tr>
                <tr bgcolor="lightgrey">
                    <td align="center">
                        <input type="submit" name="logon" value="登录">
                        <input type="reset" name="clear" value="重置">
                    </td>
                </tr>
            </table>
        </form>
```

（3）创建接收 Form 表单的页面 dologon.jsp。在该页面中对用户输入的信息进行判断，看其是否为空，如果为空则跳转到 tishi.jsp 页面显示提示信息，否则进入 welcome.jsp 欢迎页面。其关键代码如下：

```
<jsp:useBean id="mylogon" class="com.safe.UserLogon.DoyouLogon" scope="session"/>
<%
    ……//省略了获取表单数据的属性
  mylogon.setUsername(username);
  mylogon.setUserpassword(userpassword);
  String mess=mylogon.checkuser();                        //获取验证结果信息
  if(username.equals("")||userpassword.equals("")){      //如果用户输入的登录信息为空
      session.setAttribute("logonuser","");
      session.setAttribute("error",mess);
      response.sendRedirect("tishi.jsp");                 //转到提示信息页面
  }
  else{
      session.setAttribute("logonuser",mylogon);          //将用户名存入会话对象
      response.sendRedirect("welcome.jsp");               //转到欢迎页面
  }
%>
```

（4）创建一个提示页面 tishi.jsp，用来显示提示信息。在该页面中首先判断 Session 中是否存储了登录用户的信息，如果不存在则提示登录。关键代码如下：

```
<%
  String message="";
  if(session.getAttribute("logonuser")==null){           //如果用户没有登录
      message="请先<a href='index.jsp'>[登录]</a>！";
  }
  else{                                                   //如果用户登录失败
      message=(String)session.getAttribute("error");
  }
%>
    table>
          <tr bgcolor="lightgrey">
            <td align="center">友情提示！</td>
          </tr>
          <tr height="50">
            <td align="center">
              <%=message%>
            </td>
          </tr>
    </table>
```

（5）创建登录成功的欢迎页面 welcome.jsp，用于显示欢迎信息。关键代码如下：

```
<jsp:useBean id="mylogon" class="com.safe.UserLogon.DoyouLogon" scope="session"/>
<%
  String message="";
  if(session.getAttribute("logonuser")==null){
      response.sendRedirect("tishi.jsp");
  }
  else{
      message="您好！"+mylogon.getUsername()+"</b> [女士/先生]！欢迎登录！";
  }
```

```
%>
        <table>
           <tr bgcolor="lightgrey" height="30">
             <td align="center">友情提示！ </td>
           </tr>
           <tr height="50">
             <td align="center">
                 <%=message%>
             </td>
           </tr>
           </table>
```

秘笈心法

心法领悟 514：Session 共享数据。

Session 作用于同一浏览器之中，共享同一浏览器中的各个页面的数据。无论当前浏览器是否在多个页面间进行了跳转操作，整个用户会话将一直存在下去，直到关闭浏览器。

实例 515　防止页面重复提交

光盘位置：光盘\MR\20\515

中级

实用指数：★★☆

实例说明

Struts 的 Token（令牌）机制能够很好地解决表单重复提交的问题。其基本原理是：服务器端在处理客户端的请求之前，会将请求中包含的令牌值与保存在当前用户会话中的令牌值进行比较，看是否匹配。在处理完该请求后，且在将答复发送给客户端之前，将产生一个新的令牌，该令牌除传给客户端以外，也会与用户会话中保存的旧令牌进行替换。这样，如果用户回退到刚才的提交页面并再次提交，客户端传过来的令牌就和服务器端的令牌不一致，从而有效地防止重复提交。本实例的运行结果如图 20.2 所示。

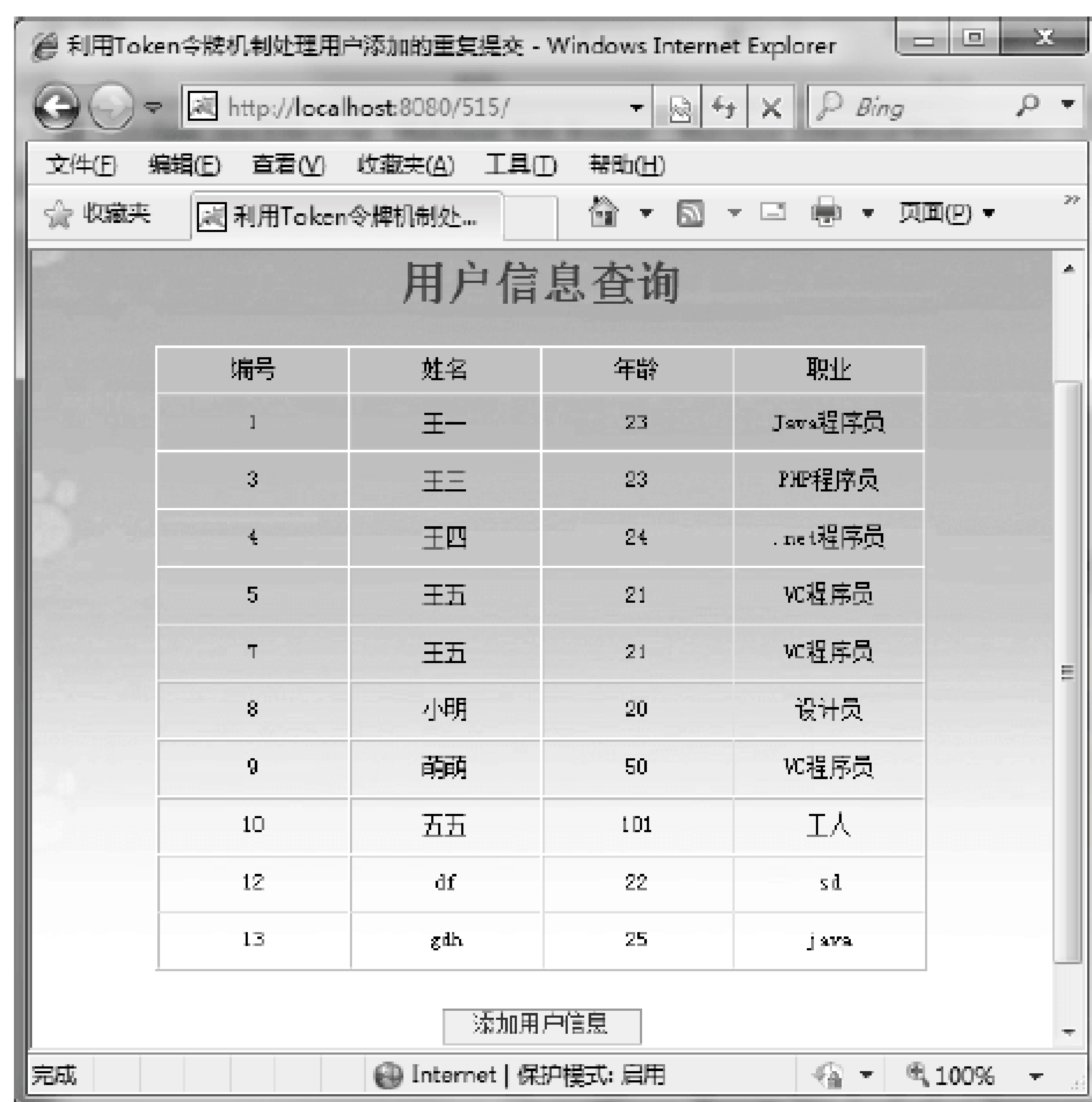


图 20.2　防止页面重复提交

本实例主要应用 Token 机制防止页面的重复提交。下面对 Token 机制的作用和语法分别进行介绍。

（1）Token 机制的作用

保存请求中的令牌值。

例如：

```
input type="hidden" name="userInfoName" value="userName"
```

value 是通过 TokenProcessor 类中的 generateToken()获得的，是根据当前用户的 Session 对象和当前时间的 long 值来计算的。

在客户端提交后，判断请求中包含的值是否和服务器的令牌一致，因为服务器每次提交都会生成新的 Token，因此如果是重复提交，客户端的 Token 值和服务器端的 Token 值就会不一致。

（2）Token 相关的方法

Struts 框架的 Token 机制在 org.apache.struts.action.Action 类中提供了一些方法。下面就以在数据库中插入一条数据来说明防止页面重复提交的主要方法。

语法：

```
protected boolean isTokenValid(javax.servlet.http.HttpServletRequest request)
```

功能：此方法判断存储在当前用户会话中的令牌值和请求参数中的令牌值是否相同，如果相同，则返回 true；如果符合以下情况之一，则会返回 false。

- 令牌值不能存在 HttpSession 对象。
- 在请求参数和 Session 会话中没有保存令牌值。
- 存储在当前用户 Session 会话中的令牌值和请求参数中的令牌值不同。

语法：

```
protected void resetToken(javax.servlet.http.HttpServletRequest request)
```

功能：从当前 Session 会话中删除令牌属性。

语法：

```
protected void saveToken(javax.servlet.http.HttpServletRequest request)
```

功能：创建一个新的令牌，并把它保存在当前 Session 范围内。如果 HttpSession 对象不存在，就首先创建一个 HttpSession 对象。

在 Action 类的 add()方法中，将 Token 值保存在页面中只需增加一条语句 saveToken(request)。其关键代码如下：

```
public ActionForward add(ActionMapping mapping, ActionForm form,HttpServletRequest request, HttpServletResponse response)
saveToken(request); //前面的处理省略
return mapping.findForward("add");
}
```

在 Action 类的 insert()方法中，将表单中的 Token 值与服务器端的 Token 值进行比较，其关键代码如下：

```
public ActionForward insert(ActionMapping mapping, ActionForm form,
HttpServletRequest request, HttpServletResponse response)
if (isTokenValid(request, true)) {
//表单不是重复提交
//这里是保存数据的代码
} else {
//表单重复提交
saveToken(request);
//其他的处理代码
}
}
```

（1）在添加用户之前，首先把请求转发给 PrepareInsertAction 类，该类将调用 saveToken(request)方法创建一个新的令牌，并把令牌保存在当前 Session 会话中；接着，PrepareInsertAction 类将把请求转发给添加用户的 insert.jsp 页面。PrepareInsertAction 类的关键代码如下：

```
public class PrepareInsertAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
                                 HttpServletRequest request,
                                 HttpServletResponse response) {
        saveToken(request);                                   //创建一个新的令牌
        return mapping.findForward("prepareInsertAction");
    }
}
```

（2）在 insert.jsp 中的<html:form>标签的处理类中判断 Session 会话中是否存在 Token，如果存在，就在表单中生成一个包含 Token 信息的隐藏字段。insert.jsp 页面中的关键代码如下：

```
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html"%>
<html:form action="userInfoAction.do">
  <table width="281" height="102" border="1">
    <tr>
      <td width="73" height="26" bgcolor="#000000"><div align="center" class="word_white">姓名</div></td>
      <td width="192"><div align="center">
      <html:text property="name"/><html:errors property="name"/>                //定义文本框表单项
      </div></td>
    </tr>
    <tr>
      <td height="32" bgcolor="#000000"><div align="center" class="word_white">年龄</div></td>
```

```
        <td><div align="center">
 <html:text property="age"/><html:errors property="age"/>
        </div></td>
      </tr>
      <tr>
        <td height="34" bgcolor="#000000"><div align="center" class="word_white">职业</div></td>
        <td><div align="center">
 <html:text property="profession"/><html:errors property="profession"/>
        </div></td>
      </tr>
   </table>
      <input type="submit" name="Submit2" value="提交">             //“提交”按钮
      <input type="reset" name="Submit" value="重置">               //“重置”按钮
<a href="index.jsp">返回</a>                                                      //“返回”超链接
   </html:form>
```

当用户收到 insert.jsp 页面后，在源文件中会看到表单中定义了一个包含 Token 信息的隐藏字段。显示的代码如下：

```
<input type="hidden" name="org.apache.struts.taglib.html.TOKEN" value="a9bf32c5fade032e405947bfe15ea18f">
```

（3）在用户提交了表单后，由 UserInfoAction 类处理请求。在 UserInfoAction 类中，首先调用 isTokenValid(request)方法，判断当前 Session 会话中的令牌值和请求参数中的令牌值是否相同。如果相同，就调用 resetToken(request)方法，从当前会话中删除 Token，然后执行添加数据的操作。UserInfoAction 类的关键代码如下：

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
                            HttpServletRequest request,
                            HttpServletResponse response) {
      UserInfoForm userInfoForm = (UserInfoForm) form;                          //获取与表单对应的 ActionForm 对象
      userInfoForm.setAge(Integer.valueOf(request.getParameter("age")));        //设置 ActionForm 对象的 age 属性
      userInfoForm.setName(Chinese.chinese(request.getParameter("name")));      //设置 ActionForm 对象的 name 属性
      userInfoForm.setProfession(Chinese.chinese(request.getParameter(
           "profession")));
      ActionMessages errors = new ActionMessages();                             //创建 ActionMessages 对象
      if (!isTokenValid(request)) {                                             //判断 Session 会话中的令牌值和请求参数中的值是否相等
         errors.add(ActionMessages.GLOBAL_MESSAGE,
                 new ActionMessage("error.invalid.token"));                     //向 ActionMessages 对象中添加对象
         saveErrors(request, errors);                                           //保存 ActionMessages 对象
         saveToken(request);                                                    //创建新的令牌
         request.setAttribute("success", "错误！！！ ");                         //将提示信息保存在 request 对象中
      }else {
         dao.addUserInfo(userInfoForm);                                         //添加用户信息的方法
         resetToken(request);
         request.setAttribute("success", "添加用户信息成功！！！ ");
      }
      return mapping.findForward("success");
   }
```

（4）在 Struts-config.xml 文件中配置 PrepareInsertAction 类和 UserInfoAction 类。关键代码如下：

```
<action-mappings>
      <action name="userInfoForm" path="/userInfoAction" scope="request" type="com.action.UserInfoAction" validate="true">
                                                                                //配置 Action
 <forward name="success" path="/success.jsp" />                                 //请求转发地址
      </action>
      <action path="/prepareInsertAction" type="com.action.PrepareInsertAction" >
          <forward name="prepareInsertAction" path="/insert.jsp"/>
      </action>
</action-mappings>
```

用户提交表单后，如果使用浏览器的后退功能退回到刚才的 insert.jsp 页面，并再次提交表单，其请求将由 UserInfoAction 类来处理，并弹出错误提示信息。

■

心法领悟 515：scope 属性。

本实例中的 scope 属性用于设置变量的作用域，其值可以是 application、request、session、page 或 action，

默认值为 action。



实例 516　对查询字符串进行 URL 编码

光盘位置：光盘\MR\20\516

中级

实用指数：★★☆

实例说明

在网站或 Web 应用程序的开发过程中，有时需要根据用户指定的查询条件来查询数据，而且这些查询条件有时会直接附加在请求的 URL 路径后作为查询参数（例如，根据商品的类别查询商品的信息）。但是有时不希望用户看到具体的查询参数信息，这时需要对查询参数字符串进行编码。运行本实例，将显示商品信息查询页面，如图 20.3 所示。用户可以输入商品类别进行查询，然后在浏览器的 URL 中附加该参数，并对其编码。

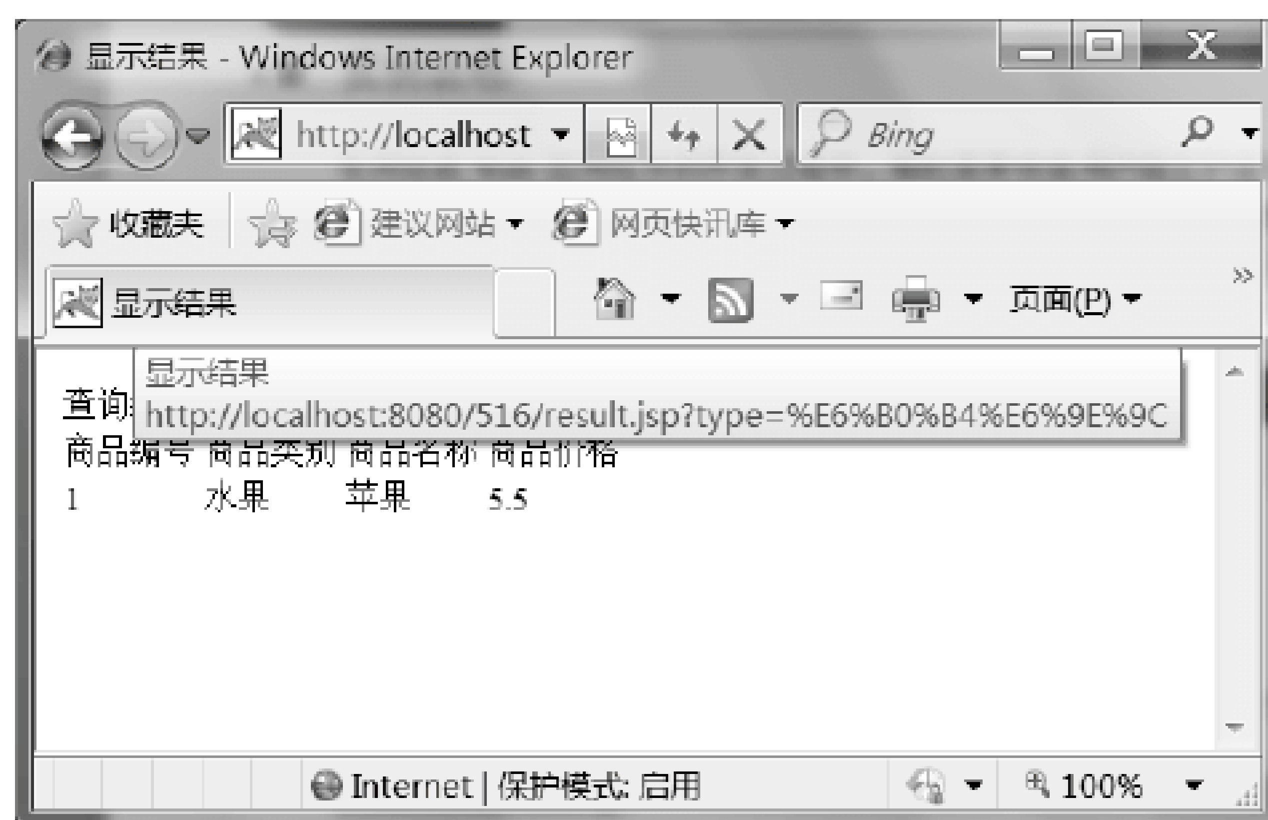


图 20.3　对查询字符串进行 URL 编码

关键技术

对 URL 进行编码时，可以通过 JavaScript 中的 encodeURI()方法实现，此方法可以将文本字符串编码为一个有效的统一资源标识符（URI）。语法如下：

```
encodeURI(URIString)
```

参数说明

URIString：字符串类型，表示要编码的字符串。

需要注意的是，encodeURI()方法不会对“:”、“/”、“;”和“?”等字符进行编辑，可使用 encodeURIComponent()方法对这些字符进行编码。

设计过程

（1）创建 index.jsp 页并添加表单，在表单中添加一个商品类别的文本框。具体代码如下：

```
<form id="selectForm">
        请输入商品类别：<input type="text" name="type" id="type">
        <input type="button" value="查询" onclick="fun()">
  </form>
```

（2）编写 JavaScript 的 fun()函数，通过 encodeURI()函数对商品类别进行编码，然后提交表单。代码如下：

```
<script type="text/javascript">
    function fun(){
        var type = document.getElementById("type").value;
        var typeParam = encodeURI(type);
        document.getElementById("selectForm").action="result.jsp";
        document.getElementById("selectForm").submit();
    }
 </script>
```

（3）创建 result.jsp 页，通过 java.net.URLDecoder 的 decode()方法对获取的请求参数解码。代码如下：

```
<%
    request.setCharacterEncoding("UTF-8");
    String goodsType = request.getParameter("type");
    goodsType = java.net.URLDecoder.decode(goodsType, "UTF-8");
    ……//省略了其他相关代码
%>
```

心法领悟 516：字符串解码。

在请求服务器将 URL 的参数提交给 Servlet 后，需要应用 java.net.URLDecoder 的 decode()方法对编码过的

参数进行解码。

实例 517　过滤非法字符

光盘位置：光盘\MR\20\517

中级

实用指数：★★★★☆

实例说明

在一些大型网站中，非法文字（如色情关键字、脏话等）过滤功能是不可或缺的，以免产生不良影响。在用户发布信息时，首先需要对发布的内容进行过滤。但网站中提交的请求很多，如果对每个请求都加入过滤代码来实现，未免过于繁琐。在本实例中，将使用 Servlet 过滤器对所有请求进行非法文字过滤。

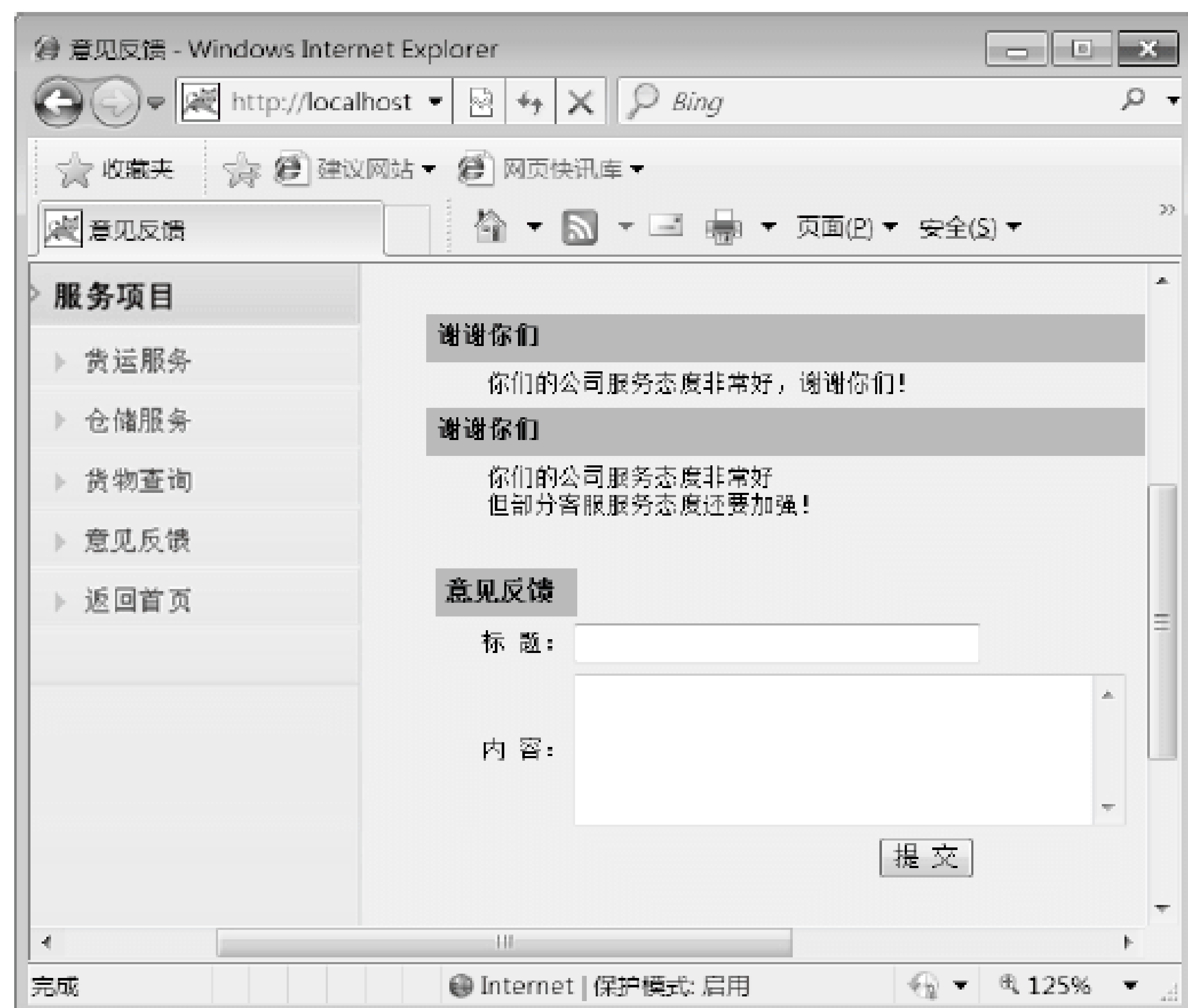


图 20.4　过滤 HTML 非法字符

关键技术

本实例中实现了 Filter 接口，作为非法文字的过滤器。此过滤器的功能强大，不仅可对非法文字进行过滤处理，还可对字符编码的转换进行处理。因此，在相关类中定义非法字符数组属性 words 与字符编码属性 encoding，并在过滤器的初始化方法 init()中对其进行实例化。其关键代码如下：

```
public class WordFilter implements Filter {
	//非法字符数组
	private String words[];
	//字符编码
	private String encoding;
	//实现 Filter 接口 init()方法
	@Override
	public void init(FilterConfig filterConfig) throws ServletException {
		//获取字符编码
		encoding = filterConfig.getInitParameter("encoding");
		//初始化非法字符数组
		words = new String[]{"糟糕","混蛋"};
	}
	//省略其他代码
}
```

（1）在过滤器的 doFilter()方法中，将传递的 ServletRequest 对象转换为自定义的对象 Request，即可实现非法字符的过滤。其关键代码如下：

```
	//实现 Filter 接口的 doFilter()方法
	@Override
```

```
    public void doFilter(ServletRequest request, ServletResponse response,
                FilterChain chain) throws IOException, ServletException {
        //判断字符编码是否有效
        if (encoding != null) {
            //设置 request 字符编码
            request.setCharacterEncoding(encoding);
            //将 request 转换为重写后的 Request 对象
            request = new Request((HttpServletRequest) request);
            //设置 response 字符编码
            response.setContentType("text/html; charset="+encoding);
        }
            chain.doFilter(request, response);
    }
```

最后通过 destroy()方法释放过滤器中的资源。其关键代码如下：

```
//实现 Filter 接口的 destroy()方法
@Override
public void destroy() {
        this.words = null;
        this.encoding = null;
}
```

（2）创建处理用户留言反馈的 Servlet 对象 MessageServlet 类，此类使用 doPost()方法对用户留言信息进行处理。其关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        //获取标题
        String title = request.getParameter("title");
        //获取内容
        String content = request.getParameter("content");
        //将标题放置到 request 中
        request.setAttribute("title", title);
        //将内容放置到 request 中
        request.setAttribute("content", content);
        //转发到 result.jsp 页面
        request.getRequestDispatcher("index.jsp").forward(request, response);
}
```

（3）对 Servlet 以及过滤器进行统一配置，其配置信息将写入 web.xml 文件中。其关键代码如下：

```
<!-- Servlet 配置  -->
<servlet>
        <servlet-name>MessageServlet</servlet-name>
        <servlet-class>com.lyq.MessageServlet</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>MessageServlet</servlet-name>
        <url-pattern>/MessageServlet</url-pattern>
</servlet-mapping>
<!--  过滤器配置  -->
<filter>
        <filter-name>WordFilter</filter-name>
        <filter-class>com.lyq.WordFilter</filter-class>
        <init-param>
                <param-name>encoding</param-name>
                <param-value>GBK</param-value>
        </init-param>
</filter>
<filter-mapping>
        <filter-name>WordFilter</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

心法领悟 517：过滤非法字符的作用。

对一些非法字符的过滤可以显著地提高网站的安全性，保护网站所在的服务器，还可以提高网站网页的布局的稳定性，在开发时一定要注意。

实例 518　禁止用户输入敏感字符

光盘位置：光盘\MR\20\518

中级

实用指数：★★★

实例说明

用户登录模块可以用来验证用户身份的合法性。但是当用户输入某些非法字符时同样会登录到系统当中，所以对此类危险字符的处理是非常有必要的。本实例将介绍如何在客户端对用户输入的内容进行验证。本实例的运行结果如图 20.5 所示。

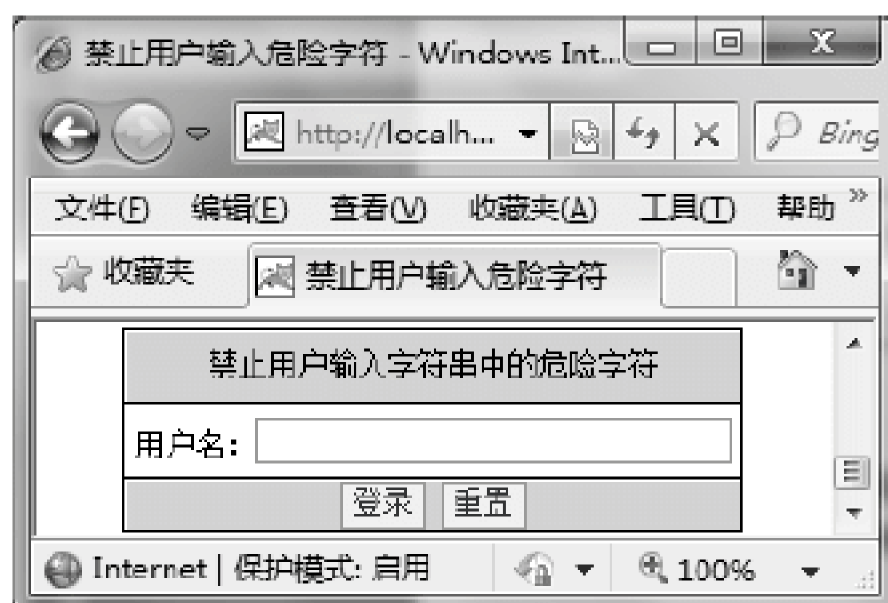


图 20.5　禁止用户输入敏感字符

关键技术

使用 JavaScript 脚本语言中的 indexOf()方法查找字符串中是否包含指定字符。

语法格式如下：

```
str.indexOf(substr)
```

功能：从字符串 str 的左面开始查找，并返回第一次出现字符串 substr 的位置；若没有找到，返回值小于 0。

参数说明

❶ str：在该字符串中进行查找。

❷ substr：要查找的字符串。

查找字符串的另一个方法的语法格式如下：

```
str.lastIndexOf(substr)
```

功能：从字符串 str 的右面开始查找，并返回第一次出现字符串 substr 的位置；若没有找到，返回值小于 0。

（1）创建登录页面 index.jsp，在该页面中要求用户输入用户名。具体代码如下：

```
<form name="form1" onsubmit="return check(form1.username.value)">
  <table>
      <td align="center">禁止用户输入字符串中的危险字符</td>
    </tr>
    <tr height="30">
      <td align="center">
        用户名：<input type="text" name="username" size="30" >
    </tr>
    <tr bgcolor="lightgrey">
      <td align="center">
        <input type="submit" name="logon" value="登录">
        <input type="reset" name="clear" value="重置">
      </td>
    </tr>
  </table>
</form>
```

（2）在 index.jsp 页面中编写验证字符串的 JavaScript 脚本。具体代码如下：

```
<script type="text/javascript">
 function check(str){
```

```
    var mess="不允许输入的字符：\r\n";
    var mark="yes";
    if(str.indexOf(";")>=0){
        mark="no";
        mess+=" ; ";
    }
    if(str.indexOf("&")>=0){
        mark="no";
        mess+=" & ";
    }
    if(str.indexOf("<")>=0){
        mark="no";
        mess+=" < ";
    }
    if(str.indexOf(">")>=0){
        mark="no";
        mess+=" > ";
    }
    if(str.indexOf("--")>=0){
        mark="no";
        mess+=" -- ";
    }
    if(str.indexOf("/")>=0){
        mark="no";
        mess+=" / ";
    }
    if(str.indexOf("%")>=0){
        mark="no";
        mess+=" % ";
    }
    if(str.indexOf("'")>=0){
        mark="no";
        mess+=" ' ";
    }
    if(mark=="no"){
        alert(mess);
        return false;
    }
    else return
        return true;
  }
</script>
```

秘笈心法

心法领悟 518：表单事件的触发。

在主页面 index.jsp 中，触发 JavaScript 脚本的事件必须使用 Form 表单的 onsubmit 事件。因为当用 Enter 键提交表单时，“登录”按钮的 onclick 事件不会被触发。

20.2　漏洞防护与数据加密

实例 519	文件上传漏洞 光盘位置：光盘\MR\20\519	高级 实用指数：★★★★

实例说明

在开发带有上传功能的程序时，对上传文件的格式和大小进行控制是非常关键的，直接关系到服务器的安全。本实例中将介绍一种通过 Servlet 控制上传文件格式和大小的方法。运行本实例，如果上传文件的格式和大小不正确，将弹出如图 20.6 所示的提示信息。



图 20.6　文件上传漏洞

本实例主要用到 Servlet 技术，具体步骤如下。

（1）首先通过 init()方法初始化上传文件的存储路径、大小和格式。

在 Servlet 实例化后，通过 init()方法实现初始化，其目的主要是为了让 Servlet 对象在处理客户请求前可以完成一些初始化的工作，例如，建立数据库连接、获取 Servlet 配置信息等。

在 Servlet 生命周期中，该方法仅被调用一次。init()方法通过 ServletConfig 类型的参数向 Servlet 传递配置信息，Servlet 使用 ServletConfig 对象从 Web 应用程序的配置信息中获取以“名-值对”形式提供的初始化参数。

在 Servlet 中还可以通过 ServletConfig 对象获取描述 Servlet 运行环境的 ServletContext 对象。使用该对象，Servlet 可以与其 Servlet 容器进行通信。

init()方法的语法如下：

```
public void init(ServletConfig arg0) throws ServletException
```

（2）通过 HttpServlet 编程类中提供的 doPost()方法将客户端的数据传递到服务器。在 doPost()方法中，根据 init()方法中提供的初始化参数实现对上传文件大小和格式的判断，指定文件在服务器中存储的位置。

doPost()方法的语法如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
```

（3）具体上传功能的实现，应用的是 jspSmartUpload 组件的 SmartUpload 类，其中通过 setMaxFileSize()方法控制上传文件的大小，通过 setAllowedFilesList()方法控制上传文件的类型。

SmartUpload 类用于实现文件的上传与下载操作，该类中提供的方法如下：

① 文件上传与文件下载必须实现的方法

在使用 jspSmartUpload 组件实现文件上传与下载时，必须先实现 initialize()方法。initialize()方法有 3 种形式，其语法如下：

```
public final void initialize(PageContext pageContext) throws ServletException
public final void initialize(ServletConfig request, HttpServletRequest response, HttpServletResponse config) throws ServletException
public final void initialize(ServletContext request, HttpSession response, HttpServletRequest out, HttpServletResponse application, JspWriter session)
            throws ServletException
```

通常应用第一种形式的方法，该方法中的 pageContext 参数为 JSP 的内置对象（页面上下文）。

② 文件上传使用的方法

要实现文件上传，首先应该实现 initialize()方法，然后应用下面两个方法将文件上传到服务器。

- upload()方法

实现 initialize()方法后，紧接着应用 upload()方法完成一些准备操作。语法如下：

```
public void upload() throws SmartUploadException, IOException, ServletException
```

首先调用 JSP 的内置对象 request 的 getInputStream()方法获取客户端的输入流，通过输入流的 read()方法读

取用户上传的所有文件数据到字节数组中，然后通过循环语句从字节数组中提取每个文件的数据，并将当前提取出的文件数据封装到 File 类对象中，最后将 File 类对象通过 Files 类的 addFile()方法添加到 Files 类对象中。

- save()方法

在实现 initialize()方法和 upload()方法后，通过调用 save()方法将全部上传文件保存到指定目录下，并返回保存的文件个数。save()方法两种形式。语法如下：

```
public int save(String destPathName) throws SmartUploadException, IOException, ServletException
```

这种形式的语法等同于 save(destPathName,0)或 save(destPathName,File. SAVE_AUTO)。

```
public int save(String destPathName, int option) throws SmartUploadException, IOException, ServletException
```

因为 save()方法调用 File 类中的 saveAs()方法保存文件，所以 save()方法中的参数的使用方法与 File 类的 saveAs()方法中的参数是相同的。但在 save()方法中，option 参数指定的保存选项的可选值为 SAVE_AUTO、SAVE_VIRTUAL 和 SAVE_PHYSICAL。它们是 SmartUpload 类中的静态字段，分别表示整数 0、1 和 2。

③ 限制上传文件的方法

上述方法能够实现文件上传的功能。下面再介绍一些在 SmartUpload 类中限制上传文件和获取其他信息的主要方法。

- setDeniedFilesList()方法：用于设置禁止上传的文件。语法如下：

```
public void setDeniedFilesList(String deniedFilesList) throws SQLException, IOException, ServletException
```

参数说明

deniedFilesList：指定禁止上传文件的扩展名，多个扩展名之间以逗号分隔。若禁止上传没有扩展名的文件，以“,,”表示。

- setAllowedFilesList()方法：设置允许上传的文件。语法如下：

```
public void setAllowedFilesList(String allowedFilesList)
```

参数说明

allowedFilesList：指定允许上传文件的扩展名，多个扩展名之间以逗号分隔。若允许上传没有扩展名的文件，以“,,”表示。例如，setAllowedFilesList("txt,doc,,")表示只允许上传*.txt、*.doc 和不带扩展名的文件。

- setMaxFileSize()方法：设定允许每个文件上传的最大长度，该长度由参数 maxFileSize 指定。语法如下：

```
public void setMaxFileSize(long maxFileSize)
```

参数说明

maxFileSize：指定上传文件的大小。

- SetTotalMaxFileSize()方法：设置允许上传文件的总长度，该长度由参数 totalMaxFileSize 指定。语法如下：

```
public void setTotalMaxFileSize(long totalMaxFileSize)
```

参数说明

totalMaxFileSize：指定上传文件的总长度。

注意：上述 4 种限制上传文件和获取其他信息的方法都必须在 upload()方法之前调用；这 4 种方法都没有返回值，通过这 4 种方式对上传的文件进行判断，如果返回结果不符合这 4 种方式中参数的设置，将抛出一个错误信息。

④ 获取文件信息的方法

下面介绍在 SmartUpload 类中获取文件信息的方法。

- getSize()方法：获取上传文件的总长度。语法如下：

```
public int getSize()
```

- getFiles()方法：获取全部上传文件，以 Files 对象的形式返回。语法如下：

```
public Files getFiles()
```

（1）创建 index.jsp 页，添加 form 表单，实现文件上传。

（2）创建 FileUpload 类，继承 HttpServlet，通过 doPost()方法对表单中提交的数据进行处理。

① 通过 Servlet 中的 init()方法，初始化上传文件的大小、类型和在服务器中存储的路径。

② 在 Servlet 的 doPost()方法中实例化 jspSmartUpload 组件中的 SmartUpload 类，应用 SmartUpload 类中的 initialize()方法，通过 setMaxFileSize()方法判断上传文件的大小，通过 setAllowedFilesList()方法判断上传文件的格式。

③ 应用 upload()方法和 save()方法实现文件上传，如果上传文件的大小、格式不正确，则在 catch 语句中抛出错误提示信息。其关键代码如下：

```
package com.pkh.servlet;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.jspsmart.upload.SmartUpload;
public class FileUpload extends HttpServlet {
    private String filedir = null;
    private long maxsize;
    private String types = null;
    public void init() throws ServletException {
        filedir = getInitParameter("filedir");                      //初始化上传文件的路径
        maxsize = Long.valueOf(getInitParameter("maxsize"));        //初始化上传文件大小
        types = getInitParameter("type");                           //初始化上传文件的类型
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doPost(request, response);
    }
    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        SmartUpload su = new SmartUpload();                         //实例化上传文件类
        try {
            su.initialize(this.getServletConfig(), request, response);  //应用 initialize()
            su.setMaxFileSize(maxsize);                             //限制上传文件大小
            su.setAllowedFilesList(types);                          //设置上传文件的类型
            su.upload();                                            //上传文件
            su.save(filedir);                                       //上传文件
            out.println("<script >alert('上传图片成功!'); window.location.href='index.jsp';</script>");
        } catch (Exception e) {
            if (e.getMessage().indexOf("extension") != -1) {
                out.println("<script >alert('上传文件的格式不正确!'); window.location.href='index.jsp';</script>");
            }
            if (e.getMessage().indexOf("Size exceeded for this file") != -1) {
                out.println("<script >alert('文件大小超出范围!'); window.location.href='index.jsp';</script>");
            }
            e.printStackTrace();
        }
    }
}
```

（3）在 web.xml 文件中配置 FileUpload 类。首先使用<servlet-name>与<servlet-class>标签配置 Servlet 的名称与所在的包类名；然后通过<init-param>标签设置上传文件的大小、上传文件类型和上传文件路径的初始值；最后通过<url-pattern>标签配置 Servlet 的映射路径。关键代码如下：

```
<servlet>
    <!--Servlet 的名称 -->
    <servlet-name>FileUpload</servlet-name>
    <!--Servlet 包的类名称 -->
    <servlet-class>com.pkh.servlet.FileUpload</servlet-class>
    <!--上传文件大小的初始值 -->
       <init-param>
          <param-name>maxsize</param-name>
          <param-value>2097152</param-value>
      </init-param>
        <!--上传文件类型的初始值 -->
      <init-param>
          <param-name>type</param-name>
          <param-value>JPG,jpg,gif,bmp,BMP</param-value>
      </init-param>
        <!--上传路径的值 -->
      <init-param>
          <param-name>filedir</param-name>
          <param-value>upload/</param-value>
```

```
            </init-param>
            </servlet>
    <!--Servlet 映射的范围 -->
  <servlet-mapping>
    <servlet-name>FileUpload</servlet-name>
    <url-pattern>/FileUpload</url-pattern>
  </servlet-mapping>
```

秘笈心法

心法领悟 519：pageContext 对象。

本实例中获取页面上下文的 pageContext 对象是一个比较特殊的对象，通过它可以获取 JSP 页面的 request、response、session、application、exception 等对象。pageContext 对象的创建和初始化都是由容器完成的，在 JSP 页面中可以直接使用该对象。

实例 520 防止资源被盗链下载

光盘位置：光盘\MR\20\520

高级

实用指数：★★★

实例说明

本实例实现的是计时拦截器。用户在登录页面中填写用户名，然后单击“提交”按钮，在控制台中便会显示程序运行的时间，如图 20.7 所示。

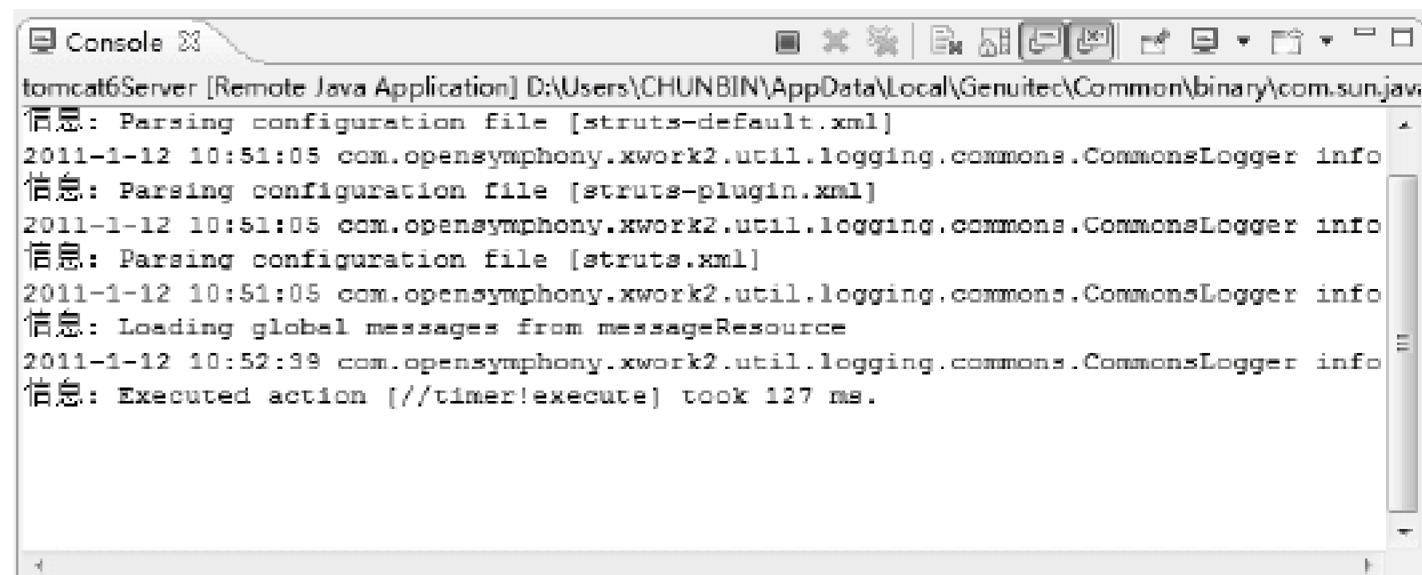


图 20.7　防止资源被盗链下载

关键技术

本实例主要通过一个随机数标识符验证下载请求是否被盗链。在生成随机标识时用到了 Math.random()方法，该方法随机生成一个 0～1 之间的 double 类型小数，将其乘以 1000 并转换成整型即可得到一个 3 位的整数。

（1）编写 index.jsp 页面文件，在该文件中首先生成一个随机数，保存在当前的用户会话信息中，然后将这个随机数赋值给页面中下载文件超链接的 id 参数。关键代码如下：

```
<%
      int d = (int)(Math.random()*1000);                    //生成 3 位整型的随机数
      session.setAttribute("d",d);                          //保存在当前会话中
%>
<table align="center" width="713" height="626" background="images/bj.jpg">
  <tr>
    <td><table style="position:relative; top:100; left:100">
        <tr>
          <td height="40"><span class="STYLE5">参赛人</span></td>
          <td height="40"><span class="STYLE5">作品名称</span></td>
          <td height="40"><span class="STYLE5">作品说明</span></td>
          <td height="40"><span class="STYLE5">作品下载</span></td>
        </tr>
        <tr>
          <td width="96" height="40">张**</td>
          <td width="124" height="40">雨后</td>
          <td width="245" height="40">刚刚下完雨后的一棵小草</td>
          <td width="77" height="40"><a href="downLoad.jsp?fileName=01.jpg&id=<%= d %>">下载</a></td>
        </tr>
        <tr>
          <td width="96" height="40">李**</td>
          <td width="124" height="40">注意危险</td>
          <td width="245" height="40">随时会掉下来哟</td>
```

```
            <td width="77" height="40"><a href="downLoad.jsp?fileName=02.jpg&id=<%= d %>">下载</a></td>
          </tr>
          <tr>
            <td width="96" height="40">唐*</td>
            <td width="124" height="40">盛开</td>
            <td width="245" height="40">盛开的一朵鲜花</td>
            <td width="77" height="40"><a href="downLoad.jsp?fileName=03.jpg&id=<%= d %>">下载</a></td>
          </tr>
          <tr>
            <td width="96" height="40">井**</td>
            <td width="124" height="40">一览众山小</td>
            <td width="245" height="40">山顶风光</td>
            <td width="77" height="40"><a href="downLoad.jsp?fileName=04.jpg&id=<%= d %>">下载</a></td>
          </tr>
        </table></td>
    </tr>
</table>
```

（2）编写 downLoad.jsp 页面文件，在该文件中首先判断当前用户会话中的 id 值是否与请求中的相同，如果相同则执行下载文件操作，如果不同则直接给出错误提示。关键代码如下：

```
<%@page contentType="text/html" pageEncoding="GBK"%>
<%@page import="java.io.*"%>
<html>
<body>
        <%
        if (session.getAttribute("d") != null&& request.getParameter("id").equals(session.getAttribute("d").toString())) {
                out.print(request.getServletPath());
                String fileName = request.getParameter("fileName");        //获取选中的文件名
                String path = request.getRealPath("/") + fileName;          //获取文件的绝对路径
                out.clear();
                out = pageContext.pushBody();
                response.setContentType("application/x-download");         //设置响应类型
                response.setHeader("Content-Disposition", "filename="+ fileName);
                InputStream in = new FileInputStream(path);                 //下载文件
                OutputStream outs = response.getOutputStream();             //创建输出流
                byte[] buf = new byte[1024];
                int len = 0;
                while ((len = in.read(buf)) > 0) {
                        outs.write(buf, 0, len);
                }
                outs.close();
                in.close();
        } else {
                %>
                请不要通过非法链接下载此文件
                <%
                        }
                %>
        </body>
</html>
```

秘笈心法

心法领悟 520：只允许注册用户下载。

防盗链下载对于一些会员制的网站来说非常重要，开发时既要防止非注册用户的盗链下载，也要注意为注册用户提供良好的下载链接。

实例 521	对登录密码进行加密 光盘位置：光盘\MR\20\521	高级 实用指数：★★★

实例说明

为了提高安全性，大多数网站都设置了用户登录模块，这样用户必须输入正确的用户名和密码后才可以进

入系统内部进行操作。本实例讲解的用户安全登录中加入了防止 SQL 注入的功能（主要是通过过滤用户所输入字符串中的危险字符实现的）。运行本实例，输入用户名和密码后，单击“登录”按钮，即可判断用户的登录状况，如图 20.8 所示。



图 20.8　对登录密码进行加密

关键技术

本实例主要用到 String 类的 replaceAll()方法。语法如下：

```
replaceAll(String rel,String with)
```

功能：替换字符串中指定的子字符串。

参数说明

❶ rel：表示要被替换的子字符串。

❷ with：表示用来替代的字符串。

设计过程

（1）创建对数据库进行操作的 DB 类文件。其中 getCon()方法用来创建一个数据库连接，getStm()方法用于获得一个 Statement 对象来执行 SQL 语句，search()方法用于对数据库进行查询操作。DB 类的关键代码如下：

```
package com.safe.UserLogon;
import java.sql.*;
public class DB {
    private Connection con;
    private Statement stm;
    private ResultSet rs;
    private String classname="com.microsoft.jdbc.sqlserver.SQLServerDriver";
    private String url="jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_database14";
    public DB(){}
    public Connection getCon(){
        try{
            Class.forName(classname);
                con=DriverManager.getConnection(url,"sa","");          //创建数据库连接
        }
        catch(Exception e){
            e.printStackTrace();
        }
        return con;                                                    //返回连接
    }
    public Statement getStm(){
        try{
            con=getCon();                                              //获取数据库连接
            stm=con.createStatement();                                 //创建 stm 对象
        }catch(Exception e){e.printStackTrace(System.err);}
        return stm;
    }
    public ResultSet search(String sql){
        if(sql==null)sql="";
         try{
             stm=getStmed();
             rs=stm.executeQuery(sql);                                 //执行 SQL 语句，获取结果集
        }
        catch(Exception e){e.printStackTrace();}
        return rs;
    }
}
```

（2）创建 Checkstr 类文件对用户输入的字符串进行过滤，具体代码如下：

```
package com.safe.UserLogon;
public class Checkstr {
    public Checkstr(){}
    public String dostring(String str){                                 //替换危险字符
        str=str.replaceAll("&","&amp;");
        str=str.replaceAll("<","&lt;");
        str=str.replaceAll(">","&gt;");
        str=str.replaceAll("'","");
        str=str.replaceAll(";","");
        str=str.replaceAll("--","");
        str=str.replaceAll("/","");
        str=str.replaceAll("%","");
        return str;
    }
}
```

（3）创建一个 Logon 类文件，在该类中调用 Checkstr 和 DB 两个类中的方法来验证用户的身份。其关键代码如下：

```
package com.safe.UserLogon;
import java.sql.*;
public class Logon {
    private String username;                                            //用户输入的姓名
    private String userpassword;                                        //用户输入的密码
    private Checkstr check=new Checkstr();                              //定义一个对用户输入的字符串进行过滤的类
    public Logon(){}
    public void setUsername(String username){
        this.username=check.dostring(username);                         //将过滤后的用户名赋值给类中的属性
    }
    ……//省略了属性 userpassword 的 setXXX()和 getXXX()方法
    public String checklogon(){                                         //进行身份验证的方法，返回提示信息
        String backmess="";
        boolean mark=true;
        if(this.username.equals("")){
           mark=false;
           backmess+="<li>请输入<b>用户名！</b></li><br>";
        }
        if(this.userpassword.equals("")){
           mark=false;
           backmess+="<li>请输入<b>密  码！</li></b>";
        }
        if(!mark){
           return backmess;
        }
        String sql="select * from tb_userlogon where user_name='"+this.username+
                                         "' and user_password='"+this.userpassword+"'"; //生成 SQL 语句
        DB db=new DB();                                                 //实例化数据库操作类
        ResultSet rs=db.search(sql);                                    //执行 SQL 语句，获取结果集
        try{
            if(rs.next()){
                backmess="登录成功！<br>用户名："+this.username+"<br>
                          密  码："+this.userpassword;
            }
            else{
                backmess="登录失败！<br>输入的<b>用户名</b>或<b>密码</b>不正确！";
            }
        }
        catch(Exception e){
            e.printStackTrace();
            backmess="操作失败！";
        }
        return backmess;
    }
}
```

（4）创建首页 index.jsp，供用户输入登录信息。其关键代码如下：

```
<form action="dologon.jsp">
 <table>
    <tr height="50">
      <td align="center">
        用户名：<input type="text" name="username" size="30">
```

```
            <br>
            密  码：<input type="password" name="userpassword" size="30">
          </td>
        </tr>
        <tr bgcolor="lightgrey">
          <td align="center">
            <input type="submit" name="logon" value="登录">
            <input type="reset" name="clear" value="重置">
          </td>
        </tr>
      </table>
    </form>
```

（5）创建接收 Form 表单的页面 dologon.jsp，该页面用于调用 Logon 类中的方法进行身份验证。其关键代码如下：

```
<jsp:useBean id="mylogon" class="com.safe.UserLogon.Logon"/>
<%
  ……                                          //省略了获取表单数据的代码
  mylogon.setUsername(username);               //设置用户输入的用户名
  mylogon.setUserpassword(userpassword);       //设置用户输入的密码
%>
      <table>
        <tr bgcolor="lightgrey" height="30">
          <td align="center">登录状况</td>
        </tr>
        <tr height="50">
          <td align="center">
            <%=mylogon.checklogon()%>           //调用验证用户身份的方法，返回提示信息
          </td>
        </tr>
      </table>
```

秘笈心法

心法领悟 521：更换 JTDS 驱动。

在连接 SQL Server 数据库时，有时会遇到 SQL Server 驱动包程序报错的情况。这时完全可以更换 JTDS 驱动：首先下载一个 jtds 1.2 以上的架包引入到项目中，然后更改连接数据库的两处代码（原 Driver（驱动）改为“net.sourceforge.jtds.jdbc.Driver;”；JDBC 改为“jdbc:jtds:sqlserver://Localhost:1433;数据库名称”），重新运行即可正常连接。

实例 522 字符串加密

光盘位置：光盘\MR\20\522

中级

实用指数：★★★

SHA 的全称为 Secure Hash Algorithm（安全散列算法），是一种不可逆的数据加密算法，现在已经成为公认的最安全的散列算法之一，并已经被广泛应用。运行本实例，在如图 20.9 所示页面中输入注册信息后单击“注册”按钮，便会显示出用户填写的注册信息（其中密码为使用 SHA 加密后的结果），如图 20.10 所示。



图 20.9　字符串加密

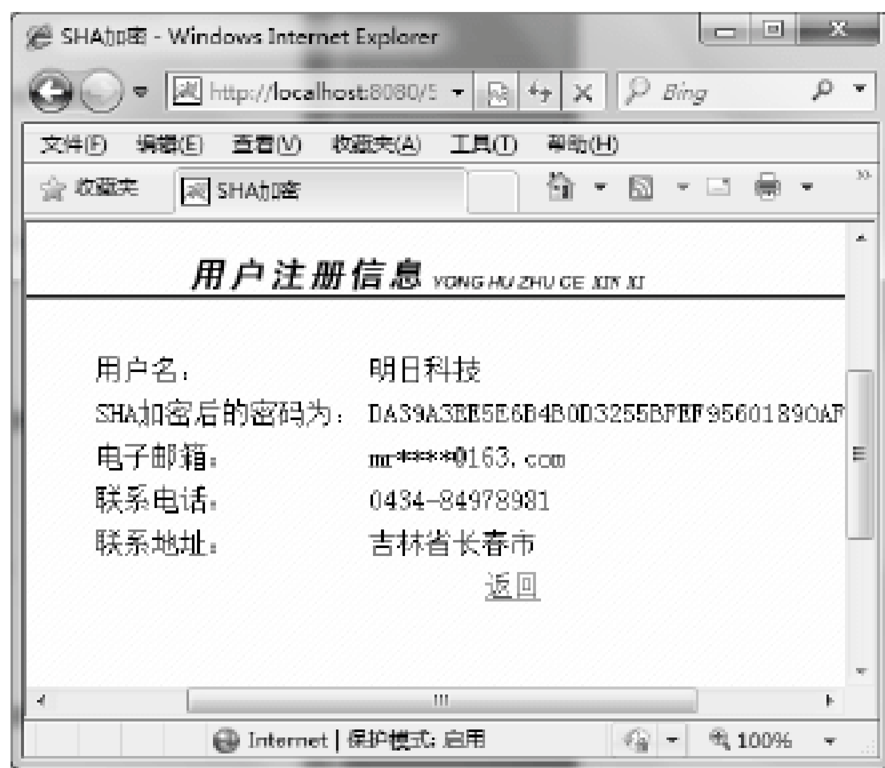


图 20.10　用户注册信息

■ 关键技术

本实例同样也用到了 java.security.MessageDigest 类，只是在构造该类对象时将构造参数设置为 SHA 即可。

■

（1）编写 MySHA.java 类文件，该类使用 SHA 算法将用户输入的字符串进行加密计算，并返回加密后的字符串。关键代码如下：

```
public class MySHA {
    public static String getMD5(String str) {
        String reStr = null;
        try {
            MessageDigest sha = MessageDigest.getInstance("SHA");        //创建具有指定算法名称的信息摘要
            sha.update(str.getBytes());                                  //使用指定的字节更新摘要
            byte ss[] = sha.digest();                                    //通过执行诸如填充之类的最终操作完成哈希计算
            reStr = bytes2String(ss);
        } catch (NoSuchAlgorithmException e) {
        }
        return reStr;
    }
    private static String bytes2String(byte[] aa) {                      //将字节数组转换为字符串
        String hash = "";
        for (int i = 0; i < aa.length; i++) {                            //循环数组
            int temp;
            if (aa[i] < 0)                                               //判断是否是负数
                temp = 256 + aa[i];
            else
                temp = aa[i];
            if (temp < 16)
                hash += "0";
            hash += Integer.toString(temp, 16);                          //转换为十六进制
        }
        hash = hash.toUpperCase();                                       //转换为大写
        return hash;
    }
}
```

（2）编写 index.jsp 页面文件，用于输入用户登录信息。关键代码如下：

```
<form name="f1" method="post" action="showInfo.jsp">
    <table style="position:relative;top:30;left:210">
        <tr>
            <td height="23">用户名：</td>
            <td height="23"><input name="name"></td>
        </tr>
        <tr>
            <td height="23">密码：</td>
            <td height="23"><input type="password" name="pwd"></td>
        </tr>
        <tr>
            <td height="23">电子邮箱：</td>
            <td height="23"><input type="text" name="mail"></td>
        </tr>
        <tr>
            <td height="23">联系电话：</td>
            <td height="23"><input type="text" name="tel"></td>
        </tr>
        <tr>
            <td height="23">联系地址：</td>
            <td height="23"><input type="text" name="addr"></td>
        </tr>
        <tr>
            <td height="23" colspan="2" align="center"><input type="submit" value="注册">
                   
                <input type="submit" value="重置"></td>
        </tr>
    </table>
```

```
</form>
```

（3）编写 showInfo.jsp 方法，获取用户输入的登录信息，并将其转换为 SHA 值，显示在页面中。关键代码如下：

```
<table style="position:relative; top:15; left:150">
    <tr>
      <td height="23">用户名：</td>
      <td height="23">${param.name }</td>
    </tr>
    <tr>
      <td height="23">SHA 加密后的密码为：</td>
      <td height="23"><%=MySHA.getMD5(pwd)%></td>
    </tr>
    <tr>
      <td height="23">电子邮箱：</td>
      <td height="23">${param.mail }</td>
    </tr>
    <tr>
      <td height="23">联系电话：</td>
      <td height="23">${param.tel }</td>
    </tr>
    <tr>
      <td height="23">联系地址：</td>
      <td height="23">${param.addr }</td>
    </tr>
    <tr>
      <td height="23" colspan="2" align="center"><a href="index.jsp">返回</a></td>
    </tr>
</table>
```

秘笈心法

心法领悟 522：什么是哈希函数。

哈希函数是一个数学方程式，可用于生成信息摘要的密码。例如，MD4、MD5、SHS 都是比较著名的哈希函数。哈希函数在现实中的信息安全方面应用较为广泛，它以哈希值“代替”信息本身，校验哈希值是否发生改变，以此判断其本身是否发生改变

实例 523 MD5 加密注册用户名和密码

光盘位置：光盘\MR\20\523

高级

实用指数：★★★

MD5 的全称为 Message-digest Algorithm5（信息摘要算法），是一种不可逆的加密算法，主要用于确保信息传输的完整一致性与系统登录认证。运行本实例，在如图 20.11 所示页面中输入注册信息后单击“注册”按钮，便会显示用户输入的注册信息（其中密码为使用 MD5 加密后的结果），如图 20.12 所示。

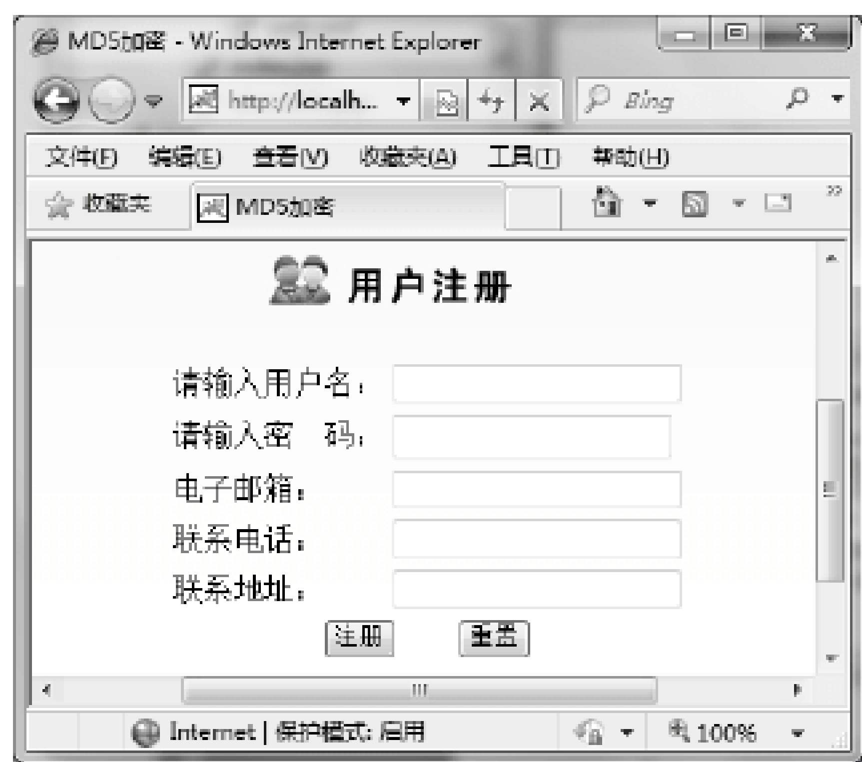


图 20.11　输入注册信息

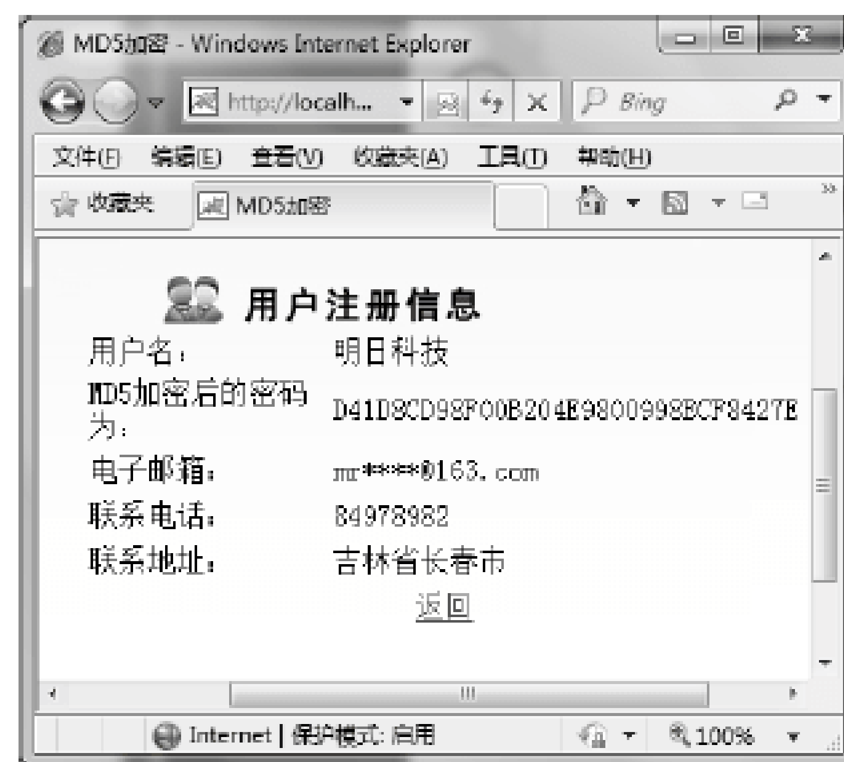


图 20.12　用户注册信息

关键技术

在将字符串转换为 MD5 值时主要使用了 java.security.MessageDigest 类，并使用 MD5 作为参数构造一个该类的对象，然后使用 update()方法更新该对象，最后通过 digest()方法完成最终的运算。

##

（1）编写 MyMD5.java 类文件，该类用于将指定的字符串转换为 MD5 值。关键代码如下：

```
public final class MyMD5 {
        public static String getMD5(String str) {
            String reStr = null;
            try {
                MessageDigest md = MessageDigest.getInstance("MD5");          //创建具有指定算法名称的信息摘要
                md.update(str.getBytes());                                    //使用指定的字节更新摘要
                byte ss[] = md.digest();                                      //通过执行诸如填充之类的最终操作完成哈希计算
                reStr = bytes2String(ss);
            } catch (NoSuchAlgorithmException e) {
            }
            return reStr;
      }
      private static String bytes2String(byte[] aa) {                        //将字节数组转换为字符串
            String hash = "";
            for (int i = 0; i < aa.length; i++) {                             //循环数组
                int temp;
                if (aa[i] < 0)                                                //如果小于 0，将其变为正数
                    temp = 256 + aa[i];
                else
                    temp = aa[i];
                if (temp < 16)
                    hash += "0";
                hash += Integer.toString(temp, 16);                           //转换为十六进制
            }
            hash = hash.toUpperCase();                                        //全部转换为大写
            return hash;
      }
}
```

（2）编写 index.jsp 表单文件，用于输入登录信息。关键代码如下：

```
<form name="f1" method="post" action="showInfo.jsp">
        <table style="position:relative; top:80; left:170">
              <tr>
                    <td height="22">请输入用户名：</td>
                    <td height="22"><input type="text" name="name"></td>
              <tr>
                    <td height="22">请输入密  码：</td>
                    <td height="22"><input type="password" name="pwd"></td>
              <tr>
                    <td height="22">电子邮箱：</td>
                    <td height="22"><input type="text" name="mail"></td>
              <tr>
                    <td height="22">联系电话：</td>
                    <td height="22"><input type="text" name="tel"></td>
              <tr>
                    <td height="22">联系地址：</td>
                    <td height="22"><input type="text" name="addr"></td>
              <tr>
                    <td height="22" colspan="2"><div align="center">
                      <input type="submit" value="注册">   
                      <input type="reset" value="重置">
                  </div></td>
            </table>
</form>
```

（3）编写 showInfo.jsp 文件，将用户输入的信息转换为 MD5 算法加密后的信息并显示在页面中。关键代码如下：

```
<%
```

```
        String pwd = request.getParameter("pwd");
%>
<table style="position:relative; top:45; left:160">
        <tr>
                <td width="130" height="23">用户名：</td>
                <td width="215" height="23">${param.name }</td>
        </tr>
        <tr>
                <td width="130" height="23">MD5 加密后的密码为：</td>
                <td width="215" height="23"><%=MyMD5.getMD5(pwd)%></td>
        </tr>
        <tr>
                <td width="130" height="23">电子邮箱：</td>
                <td width="215" height="23">${param.mail}</td>
        </tr>
        <tr>
                <td width="130" height="23">联系电话：</td>
                <td width="215" height="23">${param.tel}</td>
        </tr>
        <tr>
                <td width="130" height="23">联系地址：</td>
                <td width="215" height="23">${param.addr}</td>
        </tr>
        <tr>
                <td height="23" colspan="2" align="center"><a href="index.jsp">返回</a></td>
        </tr>
</table>
```

秘笈心法

心法领悟 523：十六进制。

十六进制（Hex Number System）是计算机数据的一种表示方法，它由 0～9 以及 A、B、C、D、E、F 组成，超过 16 则进 1。

20.3 获取客户端信息

实例 524	确定对方的 IP 地址 光盘位置：光盘\MR\20\524	高级 实用指数：★★★

实例说明

通过用户的 IP 地址可以查询到该用户的所在地及更多的信息。本实例将介绍如何获取对方的 IP 地址。运行本实例，在如图 20.13 所示页面中选中“显示”单选按钮，单击“查看”按钮，即可显示访问者计算机的 IP 地址。



图 20.13　确定对方的 IP 地址

关键技术

调用 Request 对象的 getRemoteAddr()方法可以获取客户端的 IP 地址。

语法：

```
getRemoteAddr()
```

功能：获取客户端的 IP 地址，获取值为字符串类型。

（1）创建 index.jsp 页面，用户在访问该页面时可以选择是否显示自己的 IP 地址。其关键代码如下：

```
<form action="doget.jsp">
  <table>
   <tr>
     <td align="center">
       是否显示您的 IP 地址：
       <input type="radio" name="show" value="yes" checked>显示
       <input type="radio" name="show" value="no">不显示
     </td>
   </tr>
   <tr bgcolor="lightgrey" height="25">
     <td align="center">
       <input type="submit" name="do" value="查看">
     </td>
   </tr>
  </table>
```

（2）创建接收 Form 表单的 doget.jsp 页面，该页面通过 index.jsp 页面传递的参数判断是否显示用户 IP 地址。其关键代码如下：

```
<%
  String myip="没有显示！";
  String mark=request.getParameter("show");
  if(mark==null||mark.equals("")){
     mark="no";
  }
  if(mark.equals("yes")){
     myip="您的 IP 地址为：<b>"+request.getRemoteAddr()+"</b>";          //获得用户 IP 地址
  }
%>
     <table>
       <tr>
         <td align="center">
           <%=myip%>                                                     //显示用户 IP 地址
         </td>
       </tr>
 </table>
```

秘笈心法

心法领悟 524：Request 对象。

Request 对象封装了由客户端生成的 HTTP 请求的所有细节，主要包括 HTTP 头信息、请求方式和请求参数等。通过 request 对象提供的相应方法可以获取客户端请求中的信息。

实例 525	获取客户端 TCP/IP 端口的方法 光盘位置：光盘\MR\20\525	高级 实用指数：★★★

本实例将介绍获取客户端 TCP/IP 端口的方法。运行本实例，选中“显示”单选按钮，单击“查看”按钮，将显示访问者计算机的 IP 地址及 TCP/IP 端口号。实例运行结果如图 20.14 所示。

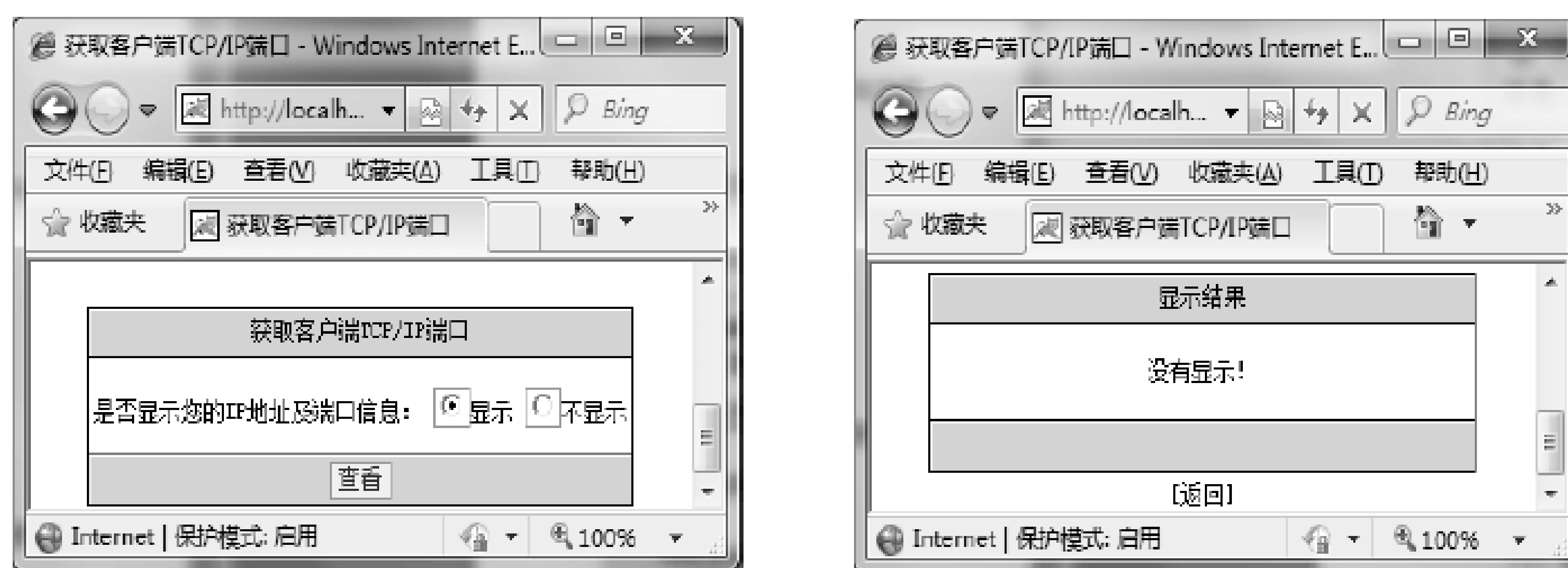


图 20.14　获取客户端 TCP/IP 端口

关键技术

调用 Request 对象的 getRemotePort()方法可以获取客户端 TCP/IP 端口。

语法：

```
getRemotePort()
```

功能：获取客户端的端口号，获取值为整型。

设计过程

（1）创建首页 index.jsp，用户在访问该页面时可以选择是否显示自己的 IP 地址及端口信息。其关键代码如下：

```
<form action="doget.jsp">
  <table>
    <tr>
      <td align="center">
         是否显示您的 IP 地址及端口信息：
         <input type="radio" name="show" value="yes" checked>显示
         <input type="radio" name="show" value="no">不显示
      </td>
    </tr>
    <tr bgcolor="lightgrey" height="25">
      <td align="center"><input type="submit" name="do" value="查看"></td>
    </tr>
  </table>
</form>
```

（2）创建接收 Form 表单的 doget.jsp 页面，该页面通过 index.jsp 页面传递的参数判断是否显示用户 IP 地址及端口信息。其关键代码如下：

```
<%
  String myip="没有显示！";
  String mark=request.getParameter("show");
  if(mark==null||mark.equals("")){
     mark="no";
  }
  if(mark.equals("yes")){
     myip="您的 IP 地址为：<b>"+request.getRemoteAddr()+"</b> ";
     myip+="端口号为：<b>"+request.getRemotePort()+"</b>";
  }
%>
     <table>
      <tr bgcolor="lightgrey" height="25">
       <td align="center">
         显示结果
       </td>
      </tr>
      <tr>
       <td align="center">
         <%=myip%>
       </td>
      </tr>
      <tr bgcolor="lightgrey" height="25">
       <td> </td>
      </tr>
 </table>
```

心法领悟 525：TCP/IP。

TCP/IP 即传输控制协议/网络通信协议，定义了电子设备（如计算机）如何连入因特网，以及数据如何在各设备之间传输的标准。

实例 526 确定对方的浏览器信息

光盘位置：光盘\MR\20\526

高级

实用指数：★★★

实例说明

现实中经常会对不同浏览器的使用数量进行统计和排名，本实例将通过 EL 表达式实现获取用户的浏览器信息，运行结果如图 20.15 所示。

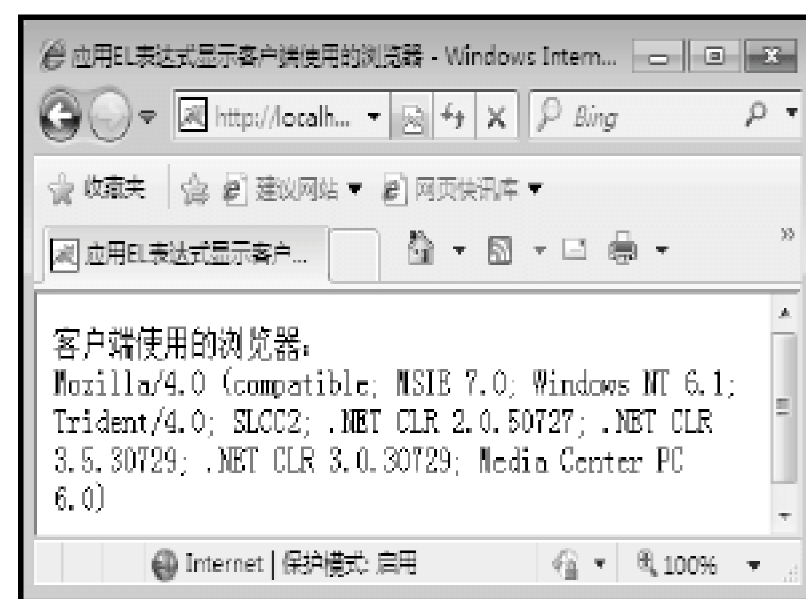


图 20.15　客户端浏览器信息

关键技术

本实例应用了 EL 中的 header 和 headerValues 对象，通过 header 对象中的 user-agent 属性值实现获取客户端使用的浏览器信息。

语法：

```
<body>
        ${header["user-agent"]}
  </body>
```

功能：获取客户端浏览器信息。

设计过程

创建 Web 项目，编写 index.jsp，在其中编写 EL 表达式，应用 header 对象，设定值为 user-agent。其关键代码如下：

```
<body>
      客户端使用的浏览器：<br>
        ${header["user-agent"]}
  </body>
```

秘笈心法

心法领悟 526：不可见的统计方式。

在现实生活中很多统计结果都是通过一些不可见的方式进行的，而这些统计中应用的相应技术也是很简单和普遍的，但是却为统计者带来了巨大的商业利润，所以一些不经意的技术往往可以起到巨大的作用。

实例 527 确定对方浏览器可接收信息的类型

光盘位置：光盘\MR\20\527

高级

实用指数：★★★

实例说明

本实例将介绍获取客户端浏览器可以接收的内容类型，运行结果如图 20.16 所示。

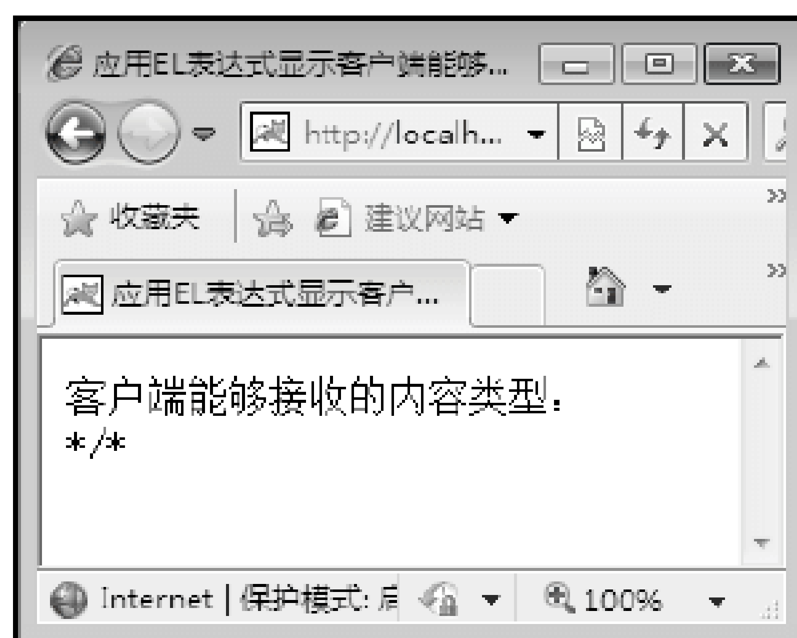


图 20.16　客户端浏览器可接收的内容类型

关键技术

应用 header 对象的 accept 属性获取客户端能够接收的内容类型。

语法：

```
<body>
        ${header["accept"]}
 </body>
```

功能：获取客户端浏览器的接收类型。

设计过程

创建 Web 项目，编写 index.jsp，在其中编写 EL 表达式，引用 header 对象，设定值为 accept。具体代码如下：

```
<body>
        客户端能够接收的内容类型：<br>
        ${header["accept"]}
 </body>
```

心法领悟 527：确定浏览器接收类型的作用。

如果没有确定好对方浏览器可以接收的类型，那么在进行数据显示时，如果传输过去的信息类型与对方浏览器可接收数据类型不匹配，就会造成接收失败而不能显示。

第21章

设计模式与架构

- 接口型模式
- 责任型模式
- 构造型模式
- 行为型模式
- 网站开发架构模式

21.1 接口型模式

实例 528

适配器模式

光盘位置：光盘\MR\21\528

初级

实用指数：★★★★☆

实例说明

什么是适配器（Adapter）设计模式？笔者先通过一个实际生活中的例子进行介绍。

例如，一般的家用电器要求的电压是 220V，有个别电器则要求使用 110V 电压，这样就必须有一个能把 220V 电压转换成 110V 电压的变压器，这个变压器就是一个适配器。

本实例将利用适配器模式实现将 220V 电压转换成 110V 电压，运行结果如图 21.1 所示。



图 21.1 适配器模式

关键技术

适配器模式把一个类的接口变换成客户端所期待的另一种接口，从而使原本接口不匹配而无法在一起工作的两个类能够在一起工作。

在以下情况下可以使用适配器模式。

- 系统需要使用现有的类，而此类的接口不符合系统的要求。
- 要建立一个可以重复使用的类，用于与该类之间关联不大的一些类，包括工作中引进的类。这些基类不一定存在复杂的接口。

类的适配器模式使用起来类似多重继承机制，利用接口的特性，把一些零散类组织到一起，成为一个新的类来实现调用，并且看起来像是对一个类的操作。实际上，适配器模式更多的是强调对代码的组织，而不是功能的实现。

通俗地讲，为了方便代码的组织与模型的准确表示，该模式在组织代码中的作用是可以把一个类中的成员插入到另一个类的继承子类中，从而让这个继承的子类看起来像一个新类，同时可以对父类减少依赖。

例如：

```
类 1:虎;类 2:鸟
```

继承一个虎，然后用适配器模式把鸟的翅膀成员拿过来，成为一个有翅膀的虎类，故有成语“如虎添翼”。如何将翅膀成员拿过来呢？这就是适配器的作用。

（1）定义适配器接口。代码如下：

```
public interface IAdapter            //适配器接口
{
 String Drive();
}
```

（2）定义适配器类 Adapter，实现 IAdapter 接口。代码如下：

```
public class Adapter implements IAdapter        //适配器类
{
 public String Drive()
        {
        return "变压器";
        }
}
```

（3）定义改变适配器类，该类主要实现将变压器的电压从 110V 改变为 220V。代码如下：

```
public class ChangeAdapter                    //改变适配器类
{
 public String Web(String str)
{
 return str;
}
}
```

（4）定义输出电压为 110V 的变压器类。代码如下：

```
public class CClass extends ChangeAdapter implements IAdapter              //实现类适配器
{
 public String Drive()
{
return this.Web("（1）输出电压：110V");
      }
}
```

（5）定义输出电压为 220V 的变压器类。代码如下：

```
public class CObject implements IAdapter        //实现对象适配器
{
 private ChangeAdapter changeAdapter;
         public CObject()
         {
 changeAdapter = new ChangeAdapter();
         }
         public String Drive()
         {
 return changeAdapter.Web("（2）输出电压：220V");
         }
}
```

（6）在 main()方法中使用变压器将 110V 的电压变压为 220V。代码如下：

```
class Program
{
      public static void main(String[] args)
{
 IAdapter dap = new Adapter();
                  System.out.println((dap.Drive()));
                  dap = new CClass();                    //调用第一个适配器
                  System.out.println((dap.Drive()));
                  dap = new CObject();                   //调用第二个适配器
                  System.out.println((dap.Drive()));
}
}
```

秘笈心法

心法领悟 528：使用适配器模式的注意事项。

适配器模式在实现时应注意以下事项：

（1）目标接口可以省略，它可以使 Adapter 不必实现不需要的方法，其表现形式就是父类实现默认方法，而子类只需实现自己独特的方法，类似模板模式。

（2）适配器类可以是抽象类。

（3）带参数的适配器模式。使用这种方法，适配器类可以根据参数返回一个合适的实例给客户端。

实例 529	外观模式 光盘位置：光盘\MR\21\529	初级 实用指数：★★★☆

实例说明

什么是外观（Facade）模式？笔者先通过一个实际生活中的例子进行介绍。

例如，打开计算机时，在计算机的内部需要执行以下几步，即启动电源、主板、硬盘，最后启动操作系统；关闭计算机时需要执行的步骤与之正好相反。启动计算机的流程如图 21.2 所示。

说明：如图 21.2 所示，用户要打开计算机，首先得启动电源，接着启动主板，然后启动硬盘，最后启动操作系统。

但是在实际生活中，要打开计算机，只需按一下电源开关即可。这里用到的正是外观模式。应用外观模式打开计算机的流程如图 21.3 所示。

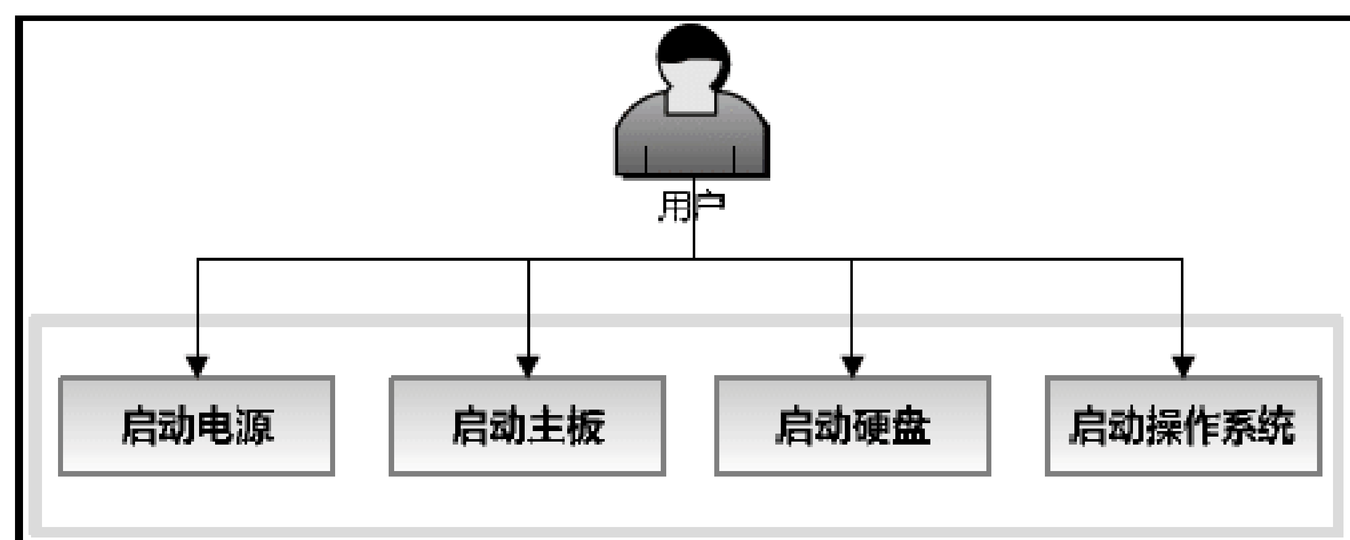


图 21.2　启动计算机的流程图

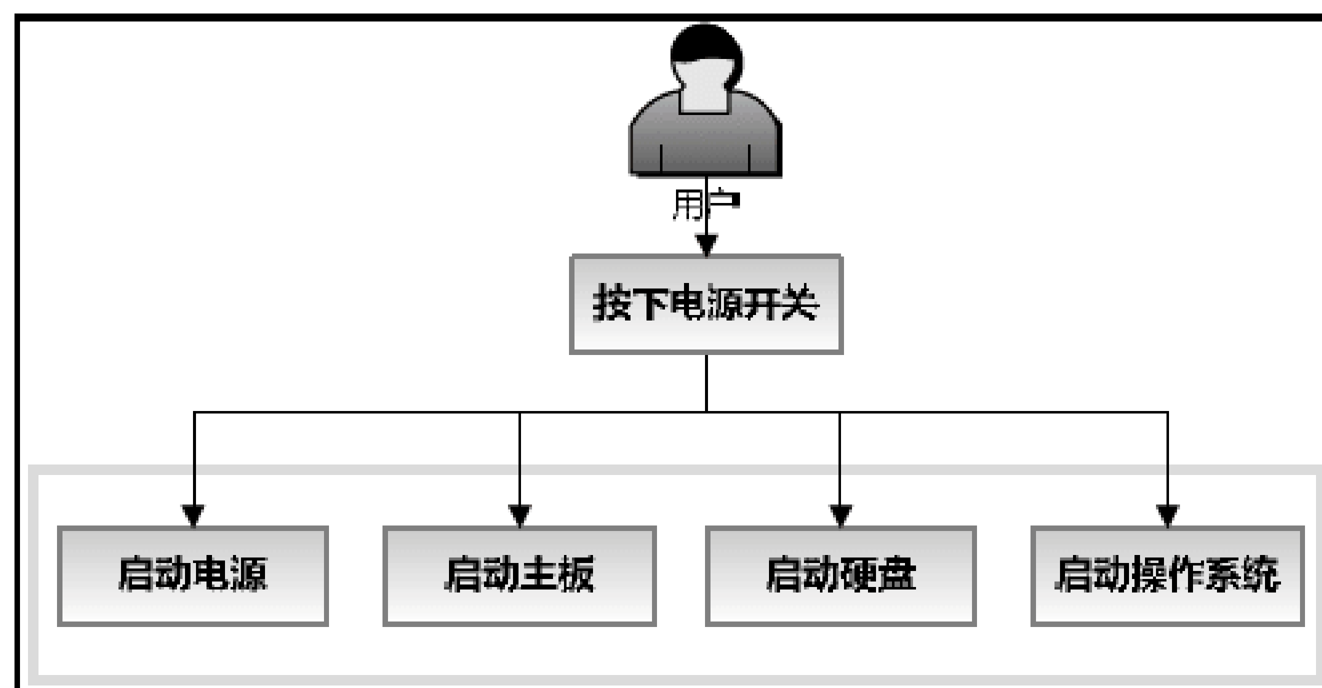


图 21.3　应用外观模式启动计算机的流程图

说明：如图 21.3 所示，用户面对的只是“按下电源开关”，至于启动计算机的具体步骤对用户来说不可见。

本实例将应用外观模式实现启动和关闭计算机，运行结果如图 21.4 所示。

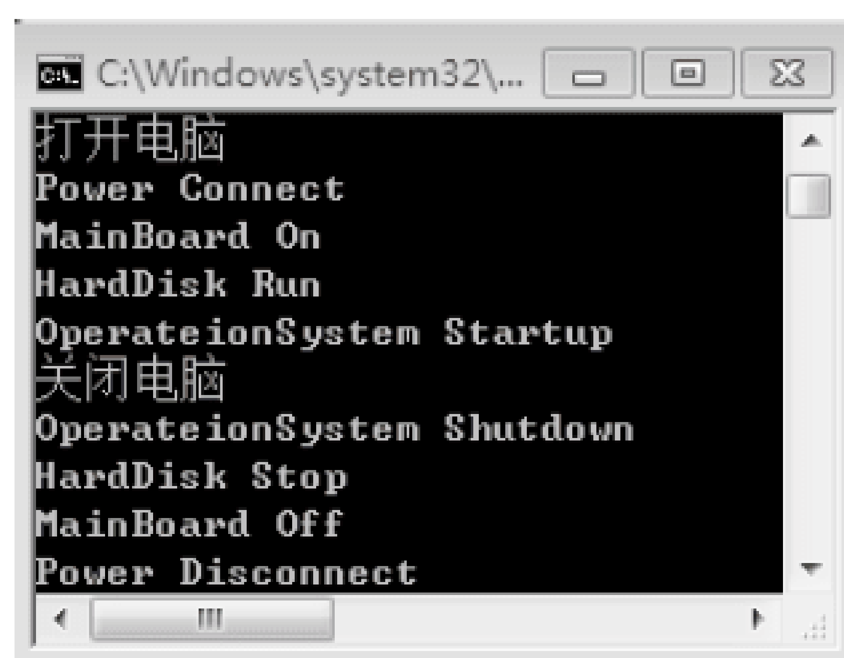


图 21.4　应用外观模式启动和关闭计算机

关键技术

下面详细介绍外观模式的意图、动机和适用性。

（1）意图

为子系统中的一组接口提供一个一致的界面。此模式定义了一个高层接口，该接口使这一子系统更易使用。

（2）动机

将一个系统划分为若干个子系统有利于降低系统的复杂性。在实际开发中，一种比较常见的设计目标是使子系统间的通信和相互依赖关系达到最小。达到该目标的途径之一是引入一个外观对象，为各个子系统提供一个单一而简单的界面。

（3）适用性

在以下情况下可以使用外观模式：

- 要为一个复杂子系统提供一个简单接口时。子系统往往因为不断演化而变得越来越复杂。大多数模式使用时都会产生更多更小的类，这使得子系统更具可重用性，也更容易对子系统进行定制，但这也给不需要定制子系统的用户带来一些使用上的困难。Facade 可以提供一个简单的默认视图，这一视图对大多数用户来说已经足够，而需要更多可定制性的用户可以越过 Facade 层。
- 客户程序与抽象类的实现部分之间存在着很大的依赖性。引入 Facade 将这个子系统与客户以及其他的子系统分离，可以提高子系统的独立性和可移植性。
- 当需要构建一个层次结构的子系统时，可以使用 Facade 模式定义子系统中每层的入口点。如果子系统之间是相互依赖的，可以使其仅通过 Facade 进行通信，从而简化它们之间的依赖关系。

设计过程

（1）定义电源类 Power，并实现启动电源和关闭电源的方法。代码如下：

```
class Power                    //电源
{
```

```
 public void Connect()
          {System.out.println("Power Connect");}
          public void Disconnect()
          {System.out.println("Power Disconnect");}
}
```

（2）定义主板类 MainBoard。代码如下：

```
class MainBoard              //主板
{
 public void On()
        {System.out.println("MainBoard On");}
        public void Off()
        {System.out.println("MainBoard Off");}
}
```

（3）定义硬盘类 HardDisk。代码如下：

```
class HardDisk               //硬盘
{
 public void Run()
          {System.out.println("HardDisk Run");}
          public void Stop()
          {System.out.println("HardDisk Stop");}
}
```

（4）定义操作系统类 OperateionSystem。代码如下：

```
class OperateionSystem             //操作系统
{       public void Startup()
        {       System.out.println("OperateionSystem Startup");
        }
        public void Shutdown()
        {       System.out.println("OperateionSystem Shutdown");
   }
}
```

（5）定义计算机类 Computer，并实现打开和关闭计算机的方法。代码如下：

```
class Computer
{       Power power;
        MainBoard mainBoard;
        HardDisk hardDisk;
        OperateionSystem operateionSystem;
        public Computer(Power power, MainBoard mainBoard, HardDisk hardDisk, OperateionSystem operateionSystem)
        {       this.power = power ;
                this.mainBoard = mainBoard ;
                this.hardDisk = hardDisk;
                this.operateionSystem = operateionSystem;
        }
        public void Startup()                //启动计算机
        {
this.power.Connect();
                this.mainBoard.On();
                this.hardDisk.Run();
                this.operateionSystem.Startup();
        }
        public void Shutdown()               //关闭计算机
        {
   this.operateionSystem.Shutdown();
                this.hardDisk.Stop();
                this.mainBoard.Off();
                this.power.Disconnect();
        }
}
```

（6）在 main()方法中，分别创建电源、主板、硬盘、操作系统及计算机类的实例，然后执行打开和关闭计算机的操作。代码如下：

```
class Program
{       public static void main(String[] args)
        {       Power power = new Power();                                        //创建电源对象
                MainBoard mainBoard = new MainBoard();                            //创建主板对象
                HardDisk hardDisk = new HardDisk();                               //创建硬盘对象
                OperateionSystem operationSystem = new OperateionSystem();        //创建操作系统对象
```

```
            Computer computer = new Computer(power, mainBoard, hardDisk, operationSystem);     //创建计算机对象
            System.out.println("打开电脑");
            computer.Startup();                                                                 //启动电脑
            System.out.println("关闭电脑");
            computer.Shutdown();                                                                //关闭电脑
    }
}
```

秘笈心法

心法领悟 529：外观模式的优点。

外观模式对客户屏蔽了子系统组件，因而减少了客户处理的对象数目，并使得子系统使用起来更加方便。

外观模式实现了子系统与客户之间的松耦合关系，而子系统内部的功能组件往往是紧耦合的。松耦合关系使得子系统的组件变化不会影响到它的客户。

外观模式有助于建立层次结构系统，也有助于对对象之间的依赖关系分层。

外观模式可以消除复杂的循环依赖关系，这一点在客户程序与子系统分别实现时尤为重要。

在大型软件系统中降低编译依赖性至关重要。在子系统类改变时，往往希望尽量减少重编译工作以节省时间。此时使用外观模式即可降低编译依赖性，限制重要系统中较小的变化所需的重编译工作。

外观模式同样也有利于简化系统在不同平台之间的移植过程，因为编译一个子系统一般不需要编译所有其他的子系统。

如果应用需要，外观模式并不限制它们使用子系统类。因此，用户可以在系统易用性和通用性之间加以选择。

实例 530 组合模式

光盘位置：光盘\MR\21\530

初级

实用指数：★★★★☆

实例说明

组合（Composite）模式有时又称部分-整体模式，它将对象组织到树形结构中，可以用来描述整体与部分的关系。组合模式可以使客户端将单纯元素与复合元素同等看待。文件夹与文件就是组合模式的典型应用，如图 21.5 所示。本实例实现的功能就是应用组合模式生成一个树形结构，运行结果如图 21.6 所示。

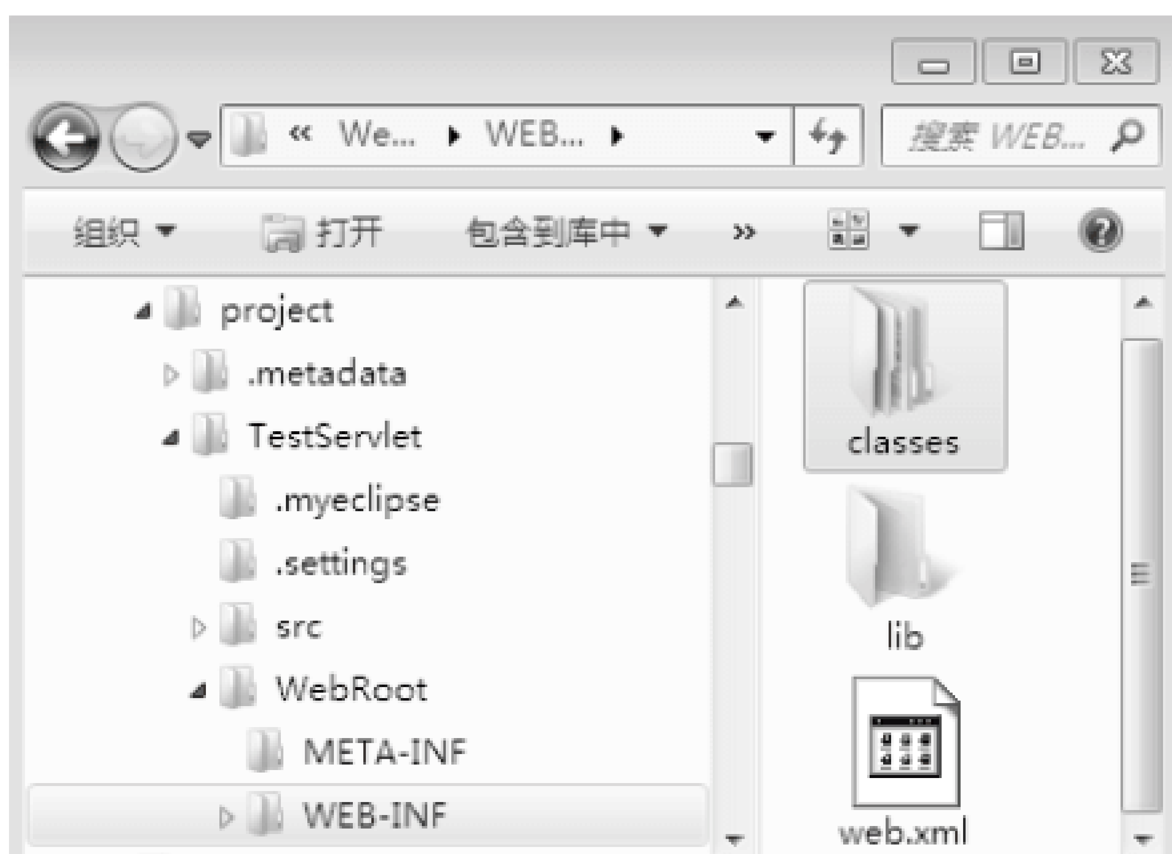


图 21.5　文件夹与文件的结构关系



图 21.6　组合模式

说明：从图 21.5 中可以看到，文件夹中还可以包含文件或其他文件夹，所以它是复合元素；而文件则是单纯元素。

关键技术

根据所实现接口的不同，组合模式可分为两种，即透明模式和安全模式。组合模式可以不提供父对象的管理方法，但必须在合适的地方提供子对象的管理方法（如 Add()方法、Remove()方法等）。

（1）透明模式

在组合结构中声明所有用来管理子类对象的方法，包括 Add()方法和 Remove()方法。这样做的优点是所有的构件类都有相同的接口。在客户端看来，树叶类对象与组合类对象的区别在接口层次上消失了，客户端可以等同地对待所有对象。这就是透明形式的组合模式。

透明模式的缺点是不够安全，因为树叶类对象和组合类对象在本质上是有所区别的。树叶类对象不可能有下一个层次的对象，因此 Add()方法、Remove()方法没有意义，但在编译时不会出错，只会在运行时出错。

（2）安全模式

在组合结构类中声明所有用于管理子类对象的方法。这样做比较安全，因为树叶类型的对象根本就没有管理子类对象的方法，因此如果客户端对树叶类对象使用这些方法，程序会在编译时出错。

安全模式的缺点是不够透明，因为树叶类和组合类将具有不同的接口。

这两种形式各有优缺点，需要根据软件的具体情况作出取舍决定。

提示：本实例中仅介绍透明模式。

（1）定义一个抽象类 AComponent 并声明相关接口。代码如下：

```
abstract class AComponent
{
protected String name;
        public AComponent(String name)
        {
System.out.println(name);
  }
        abstract public void Add(AComponent c);              //添加节点
        abstract public void Remove(AComponent c);           //移除节点
        abstract public void Display(int AComponent);        //输出节点结构
}
```

（2）定义一个类 Composite，继承抽象类 AComponent，该类用来实现组合设计模式的核心功能（本例为组合树状结构的子节点）。代码如下：

```
class Composite extends AComponent
{       static AComponent component;
      private ArrayList children = new ArrayList();

      public Composite(String name){
         super(name);
      }
      public void Add(AComponent component)
      {  this.component = component;
         children.add(component);
      }
      public void Remove(AComponent component)
      { children.remove(component); }
      public void Display(int i)
      {
         Iterator iter=children.iterator();
         AComponent ac = null;
         ac   = (AComponent)iter.next();
         while(iter.hasNext() && iter.next().equals(null))
         {
                System.out.println(ac.name);
         }
      }
}
```

（3）定义一个类 Leaf，用来向树状结构中添加子项。代码如下：

```
public class Leaf extends AComponent
{
        public Leaf(String name) {
                  super(name);
        }
```

```
        public void Add(AComponent c)
            { System.out.println("不能添加子项！"); }
        public    void Remove(AComponent c)
              { System.out.println("不能移除子项！"); }
        public    void Display(int AComponent)
              { System.out.println(new String() + name); }
}
```

（4）在 main()方法中利用透明形式的组合模式生成一个树状结构，实现代码如下：

```
public class Client
{
    public static void main(String[] args)
    {
        //建立一个树形结构
        Composite root = new Composite("根目录");              //创建一个根元素（根元素也是复合元素）
        root.Add(new Leaf("----子项  A"));
        root.Add(new Leaf("----子项  B"));
        Composite comp = new Composite("组合  X");             //创建一个复合元素
        comp.Add(new Leaf("---子项  XA"));                      //为复合元素添加单纯元素
        comp.Add(new Leaf("---子项  XB"));
        root.Add(comp);                                         //将复合元素添加到根元素
        root.Add(new Leaf("--子项  C"));                        //将单纯元素添加到根元素
        //添加和移出一个子项
        Leaf l = new Leaf("--子项  D");
        root.Add(l);
        root.Remove(l);
        //递归输出子节点
        root.Display(1);
    }
}
```

秘笈心法

心法领悟 530：使用组合模式的注意事项。

在子对象中给出明显的父对象的引用，这样可以很容易地遍历所有父对象。有了这个引用，可以方便地应用责任链模式。

通常，在系统软件里可以使用享元模式实现构件的共享，但是由于组合模式的对象经常需要引用父对象，因此共享不容易实现。

在特殊情况下，系统需要多次遍历一个树枝结构的子构件，此时应该考虑把遍历子构件的结果暂时存储在父构件中作为缓存。

关于使用什么数据类型来存储子对象的问题，在示意性的代码中使用了 ArrayList，在实际系统中可以使用其他聚集或数组等（如 List<T>）。

客户端尽量不要直接调用树叶类中的方法，而是借助其父类（Component）的多态性完成调用，这样可以增加代码的复用性。

实例 531 桥接模式

光盘位置：光盘\MR\21\531

初级

实用指数：★★★★

什么是桥接模式？笔者先通过一个实际生活中的例子进行介绍。

通讯录和游戏是每款手机必备的软件，假设手机品牌 A 和品牌 B 中都包含这两款软件，类的继承结构通常如图 21.7 或图 21.8 所示。

如果这两款手机中又增加了计算器软件或再增加一款品牌为 C 的手机，每次变化都需要添加许多类，而且需要添加的类会随变化增多，按照此种状况发展下去，就会出现类爆炸（增长为不可控制的庞然大物）。

应用桥接模式可以很好地解决这个问题。本实例运行结果如图 21.9 所示。

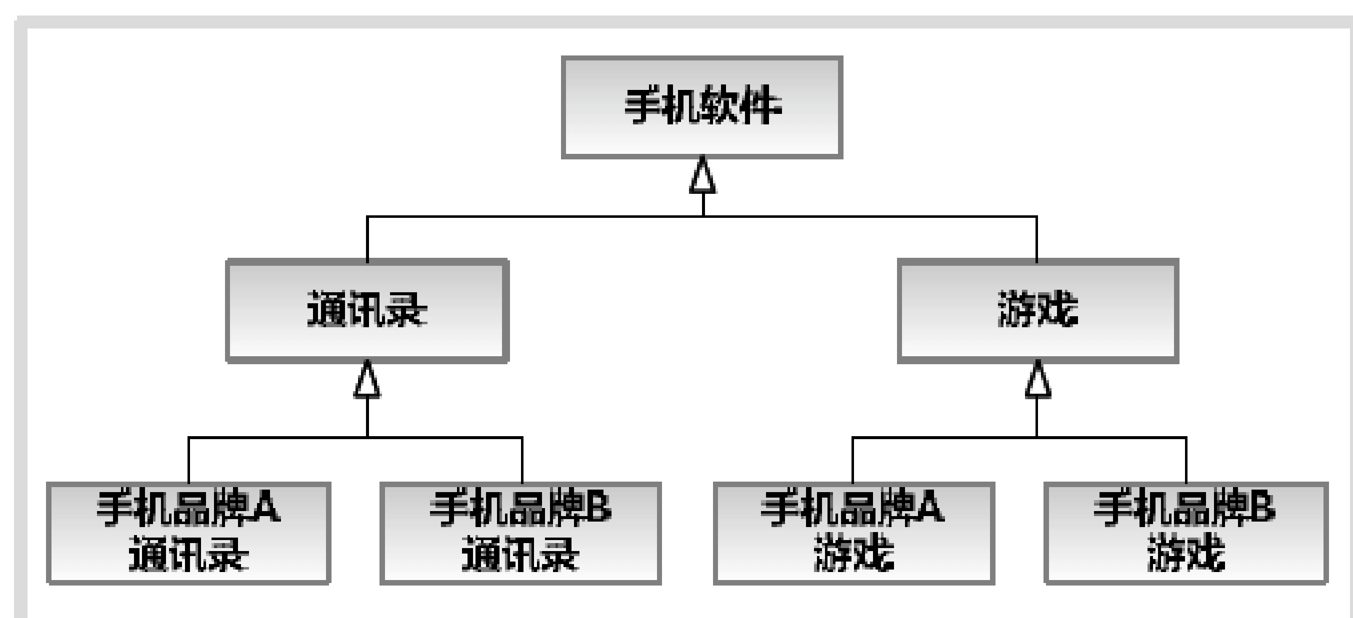


图 21.7　类的关系设计图 1

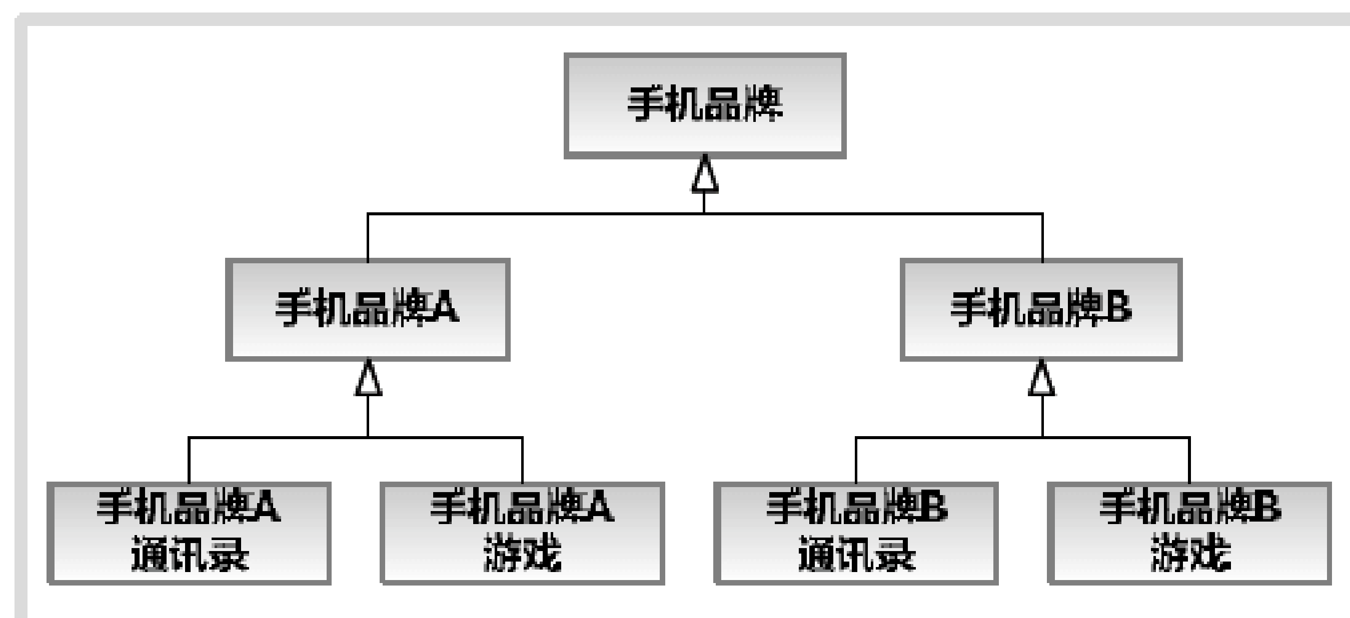


图 21.8　类的关系设计图 2

关键技术

在面向对象的设计中，有一个很重要的设计原则，那就是合成/聚合复用原则，即设计类的层次结构时优先使用对象合成/聚合，而不是类继承。

合成（Composition，也可称为组合）和聚合（Aggregation）都是关联的特殊类型。合成是一种强的“拥有”关系，体现了严格的部分和整体的关系，部分和整体的生命周期一样；聚合表示弱的“拥有”关系，体现的是 A 对象可以包含 B 对象，但 B 对象不是 A 对象的一部分。

说明： 例如，一个类中包含一个子类，类和子类之间便是合成关系；两个互不包含的平行类，当一个类中的数据成员引用另一个类的类型时，这两个类之间是聚合关系。

采用合成/聚合复用原则的优点是：优先使用对象的合成/聚合将有助于保持每个类被封装，并被集中在单个任务上。这样类和类继承层次会保持较小规模，并且不太可能出现类爆炸。本实例中对象有“手机品牌”和“手机软件”两个职责，首先分别定义“手机品牌”和“手机软件”两个抽象类，使不同的品牌和功能分别继承于这两个类，这样要增加新的品牌或新的功能时不会影响其他类。最后还要确认手机品牌和手机软件的关系，手机品牌包含有手机软件，但软件并不是品牌的一部分，所以二者之间是聚合关系。使用合成/聚合复用原则设计的类的关系如图 21.10 所示。

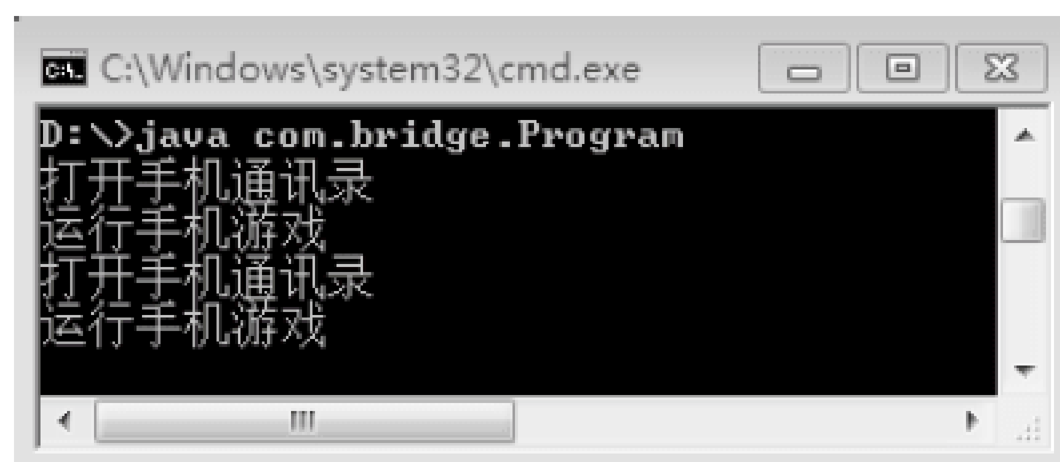


图 21.9　桥接模式

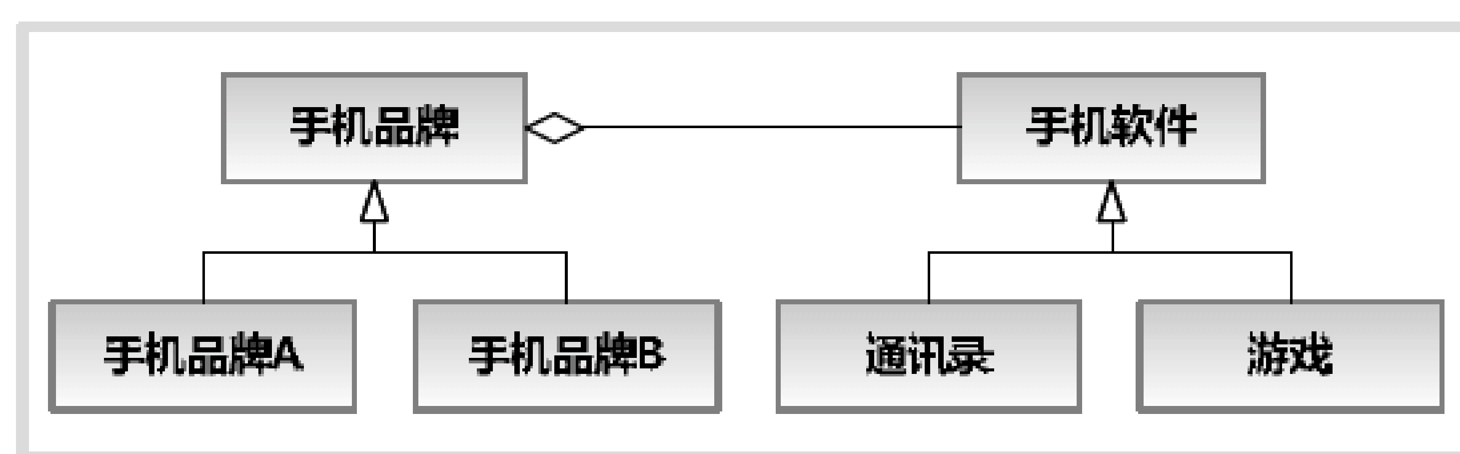


图 21.10　类的关系设计图

设计过程

（1）定义手机软件的抽象类。代码如下：

```
abstract class HandsetSoft                    //手机软件抽象类
{
    public abstract void Run();
}
```

（2）定义游戏、通讯录的具体类，继承 HandsetSoft 类，然后实现 Run()方法。代码如下：

```
class HandsetGame extends HandsetSoft          //手机游戏
{
    public void Run()
    {
        System.out.println("运行手机游戏");
    }
}
class HandsetAddressList extends HandsetSoft   //手机通讯录
{
```

```
        public void Run()
        {
            System.out.println("打开手机通讯录");
    }
}
```

（3）定义手机品牌的抽象类，其中聚合手机软件抽象类。代码如下：

```
abstract class HandsetBrand                    //手机品牌
{
    protected HandsetSoft soft;

    //设置手机软件
    public void SetHandsetSoft(HandsetSoft soft)
    {
        this.soft = soft;
    }

    public abstract void Run();
}
```

（4）定义手机品牌 A、手机品牌 B 的具体类，继承 HandsetBrand 类，然后实现 Run()方法。代码如下：

```
class HandsetBrandA extends HandsetBrand       //手机品牌 A
{
    public void Run()
    {
        soft.Run();
    }
}
class HandsetBrandB extends HandsetBrand       //手机品牌 B
{
    public void Run()
    {
        soft.Run();
    }
}
```

（5）在 main()方法中应用桥接模式，分别运行品牌 A 和品牌 B 中的通讯录和游戏。代码如下：

```
class Program
{
    public static void main(String[] args)
    {
        HandsetBrand hb;
        hb = new HandsetBrandA();
        hb.SetHandsetSoft(new HandsetAddressList());
        hb.Run();
        hb.SetHandsetSoft(new HandsetGame());
        hb.Run();
        hb = new HandsetBrandB();
        hb.SetHandsetSoft(new HandsetAddressList());
        hb.Run();
        hb.SetHandsetSoft(new HandsetGame());
        hb.Run();
    }
}
```

心法领悟 531：为手机添加计算器软件。

下面为这两款手机添加一个计算器软件。首先定义一个计算器的具体类，继承 HandsetSoft 类，然后实现 Run()方法。代码如下：

```
class HandsetCalculator extends HandsetSoft
{
    public void Run()
    {
        System.out.println("运行手机计算器");
    }
}
```

运行手机中的计算器的代码如下：

```
HandsetBrand hb;
hb = new HandsetBrandA();
hb.SetHandsetSoft(new HandsetCalculator());
hb.Run();
```

21.2　责任型模式

实例 532	单例模式 光盘位置：光盘\MR\21\532	初级 实用指数：★★★★☆

实例说明

在某种程度上，单例（Singleton）模式是限制而不是改进类的创建。单例模式可以保证一个类有且只有一个实例，并提供一个访问它的全局访问点。在程序设计过程中，有很多情况需要保证一个类只有一个实例。

单例模式的特点如下。

- 单例类只能有一个实例。
- 单例类必须自己创建其唯一实例。
- 单例类必须给所有其他对象提供这一实例。

使用单例模式需要注意以下几点。

- 使用单例模式有一个必要条件，即在一个系统要求一个类只有一个实例时才应使用单例模式；反之，如果一个类可以有几个实例共存，就不要使用单例模式。
- 不要使用单例模式存取全局变量。这违背了单例模式的用意，最好放到对应类的静态成员中。
- 不要将数据库连接做成单例模式，因为一个系统可能会与数据库有多个连接，并且在有连接池的情况下，应当尽可能及时释放连接。单例模式由于使用静态成员存储类实例，所以可能造成资源无法及时释放的问题。

本实例中实现服务器负载均衡的功能时应用了单例模式，以保证每次都使用唯一的任务分配实例挑选服务器并分配任务。实例运行结果如图 21.11 所示。



图 21.11　单例模式

说明：由于应用了单例模式，图 21.11 中显示的 4 台服务器都是同一个实例对象。

（1）在 LoadBalancer 类中创建一个类型为 LoadBalancer 的静态变量 Balancer，代码如下：

```
private static LoadBalancer Balancer;
```

（2）在 LoadBalancer 类中创建一个返回类型为 LoadBalancer 引用类型的静态方法 GetLoadBalancer()，在该方法中，首先判断静态变量 Balancer 是否为空。如果为空，调用 LoadBalancer 类的构造函数创建 LoadBalancer 类的实例对象；如果静态变量 Balancer 不为空，则在方法中直接返回该变量。代码如下：

```
public static synchronized LoadBalancer GetLoadBalancer()
    {
     if (Balancer==null)
          Balancer = new LoadBalancer();
     return Balancer;
    }
```

（3）隐藏 LoadBalancer 类的构造函数，即将构造函数的成员访问修饰符更改为 protected，强制用户只能通过 GetLoadBalancer()方法这个唯一的入口创建 LoadBalancer 类的实例对象。代码如下：

```
protected LoadBalancer()                                    //构造函数
{
```

```
        ArrayList_Server.add("服务器 I");
        ArrayList_Server.add("服务器 II");
        ArrayList_Server.add("服务器 III");
        ArrayList_Server.add("服务器 IV");
        ArrayList_Server.add("服务器 V");
}
```

设计过程

下面利用单例模式实现负载均衡实例对象。在负载均衡模型中，有多台服务器可提供服务，任务分配器随机挑选一台服务器提供服务，以确保任务均衡（实际情况要复杂得多）。这里，任务分配实例只能有一个，负责挑选服务器并分配任务。

程序主要代码如下：

```
import java.util.*;
class LoadBalancer
{
    private static LoadBalancer Balancer;
    private ArrayList ArrayList_Server = new ArrayList();
    private Random random = new Random();
    protected LoadBalancer()                                //构造函数
    {
        ArrayList_Server.add("服务器 I");
        ArrayList_Server.add("服务器 II");
        ArrayList_Server.add("服务器 III");
        ArrayList_Server.add("服务器 IV");
        ArrayList_Server.add("服务器 V");
    }
      //创建实例的唯一入口
    public static synchronized LoadBalancer GetLoadBalancer()
    {
      if (Balancer==null)
            Balancer = new LoadBalancer();
      return Balancer;
    }
    public String Server()
    {
            Object array[] = ArrayList_Server.toArray();
            return array[random.nextInt(array.length)].toString();
    }
}
public class SingletonApp
{
    public static void main(String[] args)
    {
        LoadBalancer b1 = LoadBalancer.GetLoadBalancer();    //创建类的实例
        LoadBalancer b2 = LoadBalancer.GetLoadBalancer();
        LoadBalancer b3 = LoadBalancer.GetLoadBalancer();
        LoadBalancer b4 = LoadBalancer.GetLoadBalancer();
        if ((b1 == b2) && (b2 == b3) && (b3 == b4))          //判断实例是否相同
            System.out.println("同步运行相同的实例对象");
        System.out.println(b1.Server());
        System.out.println(b2.Server());
        System.out.println(b3.Server());
        System.out.println(b4.Server());
    }
}
```

秘笈心法

心法领悟 532：最负盛名的模式。

单例模式或许是最负盛名的模式，但由于很容易误用，因此不要轻易使用。不要让单例模式成为创建全局变量的唯一方式。由于之间已经使用过模式，因此单例模式多引入的耦合并不再是相对较好的选择。应该减少操作单例模式的类的数量；最好让类知道它们正在使用的一个对象，但不必了解这个对象有哪些限制。

实例 533

建造者模式

光盘位置：光盘\MR\21\533

初级

实用指数：★★★★☆

实例说明

建造者（Builder）模式可以将一个产品的内部表象与产品的生成过程分割开来，从而可以使一个建造过程生成具有不同内部表象的产品对象。

一个对象都会有一些比较重要的性质，在没有恰当的值之前，对象不能作为一个完整的产品使用。例如，一个 E-mail 有发件人地址、收件人地址、主题、内容和附件等部分，如果收件人地址未被赋值，那么这个 E-mail 是不能发出的。

一个对象的一些性质必须按照某个顺序赋值才有意义。在某个性质没有赋值之前，另一个性质无法赋值，这样，性质本身的建造涉及复杂的商业逻辑。

因此，对象相当于一个待建造的产品，而对象的这些性质相当于产品的零件，建造产品的过程就是组合零件的过程。由于组合零件的过程很复杂，因此这些“零件”的组合过程往往被“外部化”到一个称为建造者的对象中，建造者返还给客户端的是一个全部零件都建造完毕的产品对象。

在以下情况中应当使用建造者模式。

- 需要生成的产品对象有复杂的内部结构。
- 需要生成的产品对象的属性相互依赖，建造者模式可以强迫生成顺序。
- 在对象创建过程中会用到系统中的一些其他对象，这些对象在产品对象的创建过程中不易得到。

使用建造者模式可以达到以下效果。

- 建造者模式的使用使得产品的内部表象可以独立地变化，客户端不必知道产品内部组成的细节。
- 每一个“建造者”都相对独立，而与其他的“建造者”无关。
- 使用该模式建造的最终产品更易于控制。

本实例将应用建造者模式分别建造小汽车对象和摩托车对象，并显示建造的内容，如图 21.12 所示。

```
C:\Windows\system32\cmd.exe
---------摩托车---------
框架  :          摩托车 框架
发动机 :          500 毫升
车轮数量:          2个车轮
车门数量:          没有车门
---------轿车---------
框架  :          轿车 框架
发动机 :          2500 毫升
车轮数量:          4个车轮
车门数量:          4个车门
```

图 21.12　建造者模式

应用建造者模式之前，必须知道建造者模式的优点。该模式具有以下优点。

- 可以使得产品内部的表象独立变化。在原来的工厂方法模式中，产品内部的表象是由产品自身来决定的；而在建造者模式中，“外部化”为由建造者负责。这样定义一个新的具体建造者角色就可以改变产品的内部表象，符合“开闭原则”。
- 使得客户不需要知道太多产品内部的细节。它将复杂对象的组建和表示方式封装在一个具体的建造者角色中，而且由指导者协调建造者角色得到具体的产品实例。
- 每一个具体建造者角色是毫无关系的。
- 建造者模式可以对复杂产品的创建进行更加精细的控制。产品的组成是由指导者角色调用具体建造者角色逐步完成的，所以比其他创建型模式能够更好地反映产品的构造过程。

（1）在 main()方法中分别创建工厂类、小汽车建造类和摩托车建造类的实例对象，使用工厂对象分别建造小汽车对象和摩托车对象，并显示建造的内容。代码如下：

```
public class BuilderApp
{
    public static void main(String[] args)
    {
            //创建摩托车
            MotorCycleBuilder builder1 = new MotorCycleBuilder();
            Vehicle director1 = new Vehicle(builder1);
            //创建小汽车
            CarBuilder builder2 = new CarBuilder();
        Vehicle director2 = new Vehicle(builder2);
        director1.construct();
        director2.construct();
        //显示创建产品
        builder1.shop.show();
        builder2.shop.show();
}
}
```

（2）定义交通工具建造的接口（包含 Vehicle 属性），声明实现建造过程的各个方法。代码如下：

```
interface VehicleBuilder            //建造者
{
      void BuildFrame();
      void BuildEngine();
      void BuildWheels();
      void BuildDoors();
      void Buider();
}
```

（3）定义摩托车建造类，继承交通工具建造抽象类，并实现抽象类中定义的方法。代码如下：

```
class MotorCycleBuilder implements VehicleBuilder            //具体建造者 1
{
            Shop shop = new Shop();
            public void Buider()
            {
                  shop.object("type", "摩托车");
            }
            public void BuildFrame()
            {
                  shop.object("frame","摩托车 框架");
            }
          public void BuildEngine()
          {
            shop.object("engine","500 毫升");
          }
          public void BuildWheels()
          {
            shop.object("wheels", "2 个车轮");
          }
           public void BuildDoors()
          {
            shop.object("doors","没有车门");
          }
}
```

（4）定义小汽车建造类，继承交通工具建造抽象类，并实现抽象类中定义的方法。代码如下：

```
class CarBuilder implements VehicleBuilder            //具体建造者 2
{
            Shop shop = new Shop();
            public void Buider(){
                  shop.object("type","轿车");
            }
      public void BuildFrame()
      {
```

```
            shop.object("frame","轿车 框架");
        }
        public void BuildEngine()
        {
        shop.object("engine","2500 毫升");
        }
        public void BuildWheels()
        {
            shop.object("wheels", "4 个车轮");
        }
        public void BuildDoors()
        {
            shop.object("doors","4 个车门");
        }
}
```

（5）定义交通工具类，该类主要实现记录并显示交通工具的建造内容。代码如下：

```
class Vehicle
{
            private VehicleBuilder builder;
            public Vehicle( VehicleBuilder builder )
            {
                this.builder = builder;
            }
            //这是将部件装成汽车的过程
            public void construct()
            {
                builder.BuildFrame();
                builder.BuildEngine();
                builder.BuildDoors();
                builder.BuildWheels();
                builder.Buider();
            }
}
```

（6）定义建造交通工具的工厂类，在工厂类中规定交通工具的建造步骤。代码如下：

```
class Shop
{
            String type;
             Hashtable parts = new Hashtable();
            public void object (String key, String value)
            {
                parts.put(key , value);
             }
            public void show() {
                System.out.println("--------" + parts.get("type")+"--------");
                System.out.println("框架 ：            " + parts.get("frame"));
                System.out.println("发动机 ：          " + parts.get("engine"));
                System.out.println("车轮数量：         " + parts.get("wheels"));
                System.out.println("车门数量：         " + parts.get("doors"));
            }
}
```

■ 秘笈心法

心法领悟 533：建造者模式的深入讨论。

建造者模式中需要用到组成成品的各种组件类，对于这些类的创建可以考虑使用工厂方法或者原型模式实现，在必要时也可以加上单例模式来控制类实例的产生。但前提是要使引入的模式为开发的系统带来好处，而不是臃肿的结构。建造者模式在得到复杂产品时要引用多个不同的组件，从这一点来看，它和抽象工厂模式是相似的。可以通过以下两点来区分两者：建造者模式着重于逐步将组件装配成一个成品并向外提供成品，而抽象工厂模式着重于得到产品族中相关的多个产品对象；抽象工厂模式的应用是受限于产品族的，建造者模式则不会。

由于建造者模式和抽象工厂模式在实现功能上相似，所以两者使用的环境都比较复杂并且需要更多的灵活性。建造者模式中可能要用到不同“大小”的组件类，因此这时也经常和合成模式结合使用。

实例 534　中介者模式　初级

光盘位置：光盘\MR\21\534　实用指数：★★★★☆

实例说明

什么是中介者（Mediator）模式？笔者先通过一个实际生活中的例子进行介绍。例如，卖房者和买房者一般都通过房屋中介公司进行房产交易，卖房者先到中介公司进行售房登记，买房者到中介公司查询到满意的房源信息后，通知中介公司看房，中介公司再通知卖房者……在整个房屋买卖的过程中，买房者和卖房者不直接联系，而是通过中介公司。通过中介公司进行房屋买卖是一种标准的中介者模式。

本实例中应用中介者模式实现同一公司的两位同事进行通信，运行结果如图 21.13 所示。



图 21.13　中介者模式

关键技术

应用中介者模式之前必须掌握中介者模式的概念及适用性，下面分别介绍。

（1）中介者模式的概念

用一个中介对象来封装一系列的对象交互。中介者使各对象不需要显式地相互引用，从而使其耦合松散，而且可以独立地改变它们之间的交互。本实例中类的关系如图 21.14 所示。

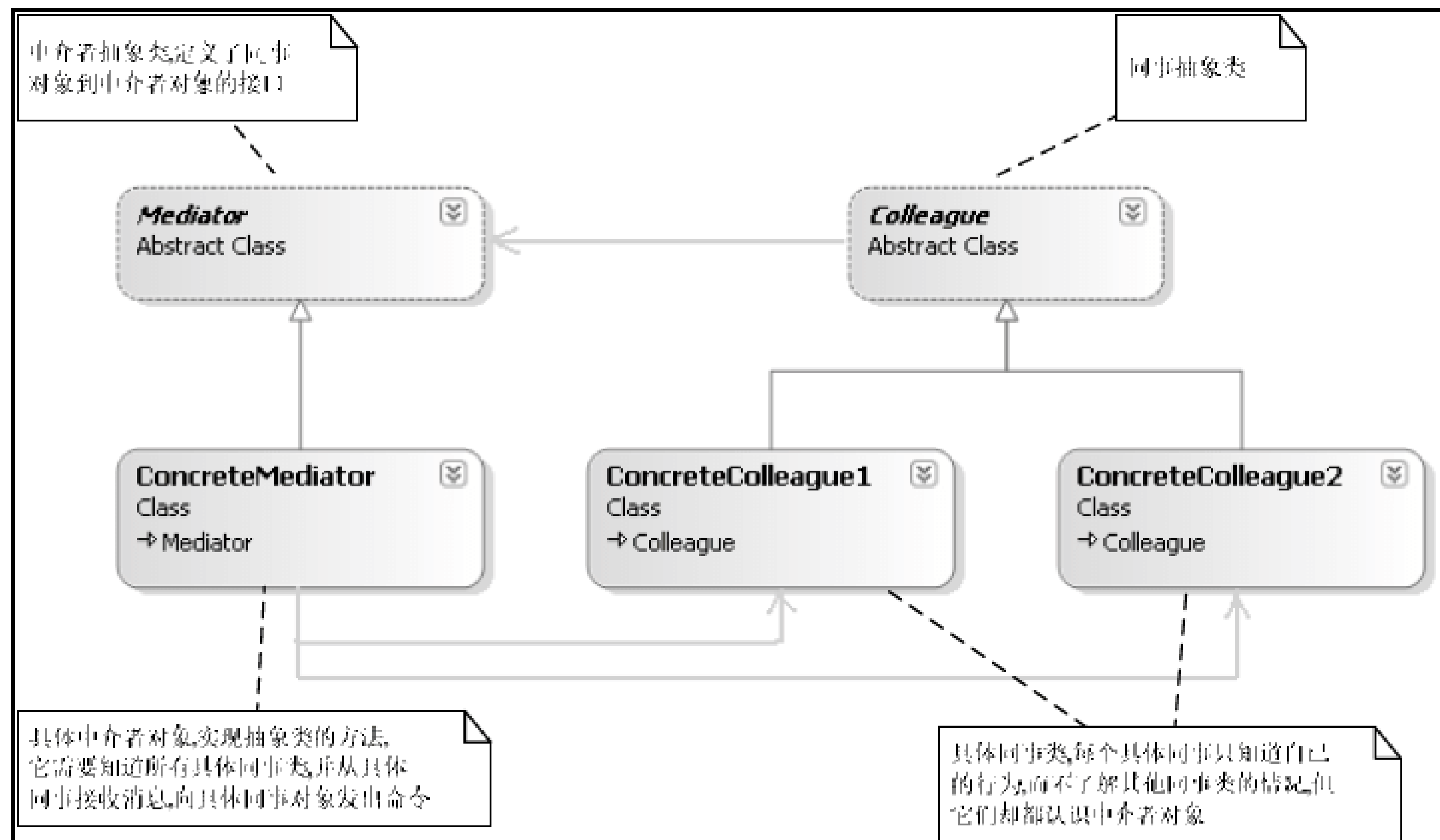


图 21.14　类关系设计图

（2）中介者模式的适用性

在下列情况下使用中介者模式。

- ❑ 一组对象以定义良好但是复杂的方式进行通信，产生的相互依赖关系结构混乱且难以理解。
- ❑ 一个对象引用其他很多对象并且直接与这些对象通信，导致难以复用该对象。
- ❑ 想定制一个分布在多个类中的行为，而又不想生成太多的子类。

（1）定义中介者的抽象类 Mediator，并声明发送消息方法 Send()。代码如下：

```
abstract class Mediator                                    //中介者抽象类
{
    public abstract void Send(String message, Colleague colleague); //发送消息的方法
}
```

（2）定义具体的中介者类 ConcreteMediator，继承 Mediator 类。代码如下：

```
class ConcreteMediator extends Mediator                    //具体的中介者类
{
     ConcreteColleague1 colleague1;
     ConcreteColleague2 colleague2;
    public void Send(String message,Colleague colleague)     //重写发送消息的方法，根据对象作出选择判断通知对象
    {
        if (colleague == colleague1)
        {
            colleague2.Notify(message);
        }
        else
        {
            colleague1.Notify(message);
        }
    }
}
```

（3）定义同事抽象类 Colleague，主要实现在构造函数中获得中介者类。代码如下：

```
abstract class Colleague                                   //同事抽象类
{
    protected Mediator mediator;
    public Colleague(Mediator mediator)                    //构造函数，得到中介者
    {
        this.mediator = mediator;
    }
}
```

（4）定义第 1 个具体的同事类，并实现发送消息和接收消息的方法。代码如下：

```
class ConcreteColleague1 extends Colleague                 //具体的同事类 1
{
    public ConcreteColleague1(Mediator mediator)
     {super(mediator);}
    public void Send(String message)
    {
        mediator.Send(message, this);                      //发送消息时通常是中介者发送出去的
    }
    public void Notify(String message)
    {
       System.out.println("同事 A 收到消息： " + message);
    }
}
```

（5）定义第 2 个具体的同事类，并实现发送消息和接收消息的方法。代码如下：

```
class ConcreteColleague2 extends Colleague                 //具体的同事类 2
{
    public ConcreteColleague2(Mediator mediator)
     {super(mediator);}
    public void Send(String message)
    {
        mediator.Send(message, this);
    }
    public void Notify(String message)
    {
        System.out.println("同事 B 收到消息：  " + message);
    }
}
```

（6）在 main()方法中，分别创建中介对象和两个同事对象，然后让两个同事对象与中介对象互相了解，最后同事 B 发送消息，同事 A 回复消息（都是通过中介对象转发）。代码如下：

```
class Program
{
    public static void main(String[] args)
    {
        ConcreteMediator m = new ConcreteMediator();       //创建具体中介者对象
```

```
        ConcreteColleague1 c1 = new ConcreteColleague1(m);      //让两个具体的同事类认识中介者对象
        ConcreteColleague2 c2 = new ConcreteColleague2(m);
        m.colleague1 = c1;                                      //让中介者对象认识同事对象
        m.colleague2 = c2;
        c1.Send("How are you?");                                //具体同事对象的发送信息都是通过中介者转发的
        c2.Send("Fine, thanks");
    }
}
```

秘笈心法

心法领悟 534：中介者模式的优点和缺点。

- 减少了子类生成。Mediator 将原本分布于多个对象间的行为集中在一起，改变这些行为只需生成 Mediator 的子类即可，这样各个 Colleague 类可被重用。
- 简化了对象协议，用 Mediator 和各 Colleague 间的一对多的交互来代替多对多的交互，更易于理解、维护和扩展。
- 对对象如何协作进行了抽象，将中介作为一个独立的概念并将其封装在一个对象中，使用户将注意力从各对象本身的行为转移到它们之间的交互上来。这有助于弄清楚一个系统中的对象是如何交互的。
- 使控制集中化。中介者模式将交互的复杂性变为中介者的复杂性。因为中介者封装了协议，它可能变得比任何一个 Colleague 都要复杂。这可能导致中介者自身成为一个难于维护的庞然大物。

实例 535 责任链模式

光盘位置：光盘\MR\21\535

初级

实用指数：★★★★☆

实例说明

什么是责任链模式？笔者先通过一个实际生活中的例子进行介绍。

例如，笔者为明日公司的一名员工，由于特殊原因需要请 7 天假。首先到 Java 部门经理处请假，部门经理只有权准许请 1 天假；于是笔者又到总监处请假，总监只有权准许请 3 天假；最后，只能到总经理处请假，结果总经理同意了。

以上的请假流程可以使用责任链模式实现，部门经理、总监、总经理都是责任链中的一环，他们都有自己的责任（如总监只有权准许下属请 3 天假），如果在自己责任范围内则可以直接处理，如果不在自己责任范围内则可以报送上级处理。

本实例主要演示应用责任链模式实现笔者的请假流程，运行结果如图 21.15 所示。

应用责任链模式之前必须掌握责任链模式的概念和适用性，下面分别介绍。

（1）责任链模式的概念

责任链模式是使多个对象都有机会处理请求，从而避免请求的发送者和接收者之间的耦合关系。将这些对象连成一条链，并沿着这条链传递该请求，直到有一个对象处理请求为止。本实例中类的关系如图 21.16 所示。

说明：在图 21.16 中，Manager 是管理者抽象类；CommonManager 是部门经理类；Majordomo 是总监类；GeneralManager 是总经理类；Request 是申请类。

（2）责任链模式的适用性

满足以下条件时可以使用责任链模式。

- 有多个对象可以处理一个请求，哪个对象处理该请求运行时自动确定。
- 想在不明确指定接收者的情况下，向多个对象中的一个提交一个请求。
- 可处理一个请求的对象集合应被动态指定。

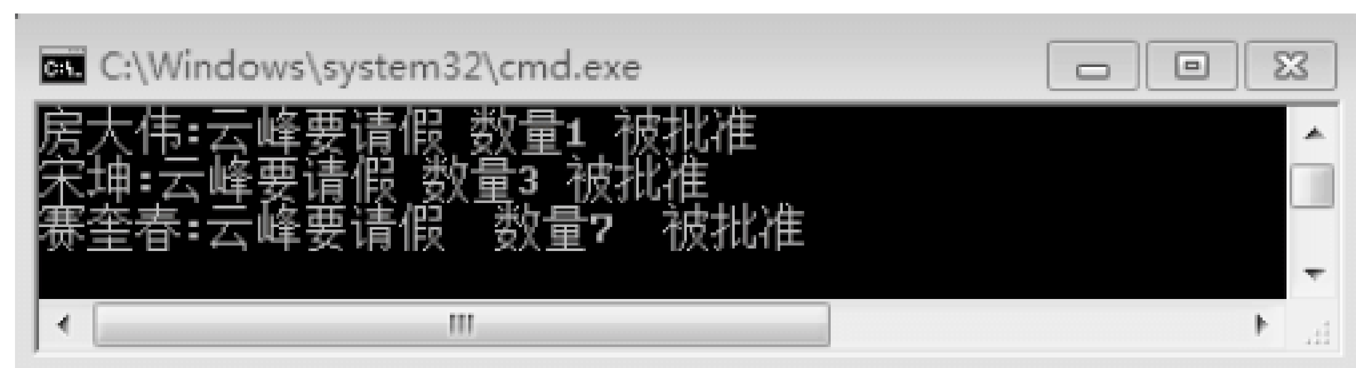


图 21.15　责任链模式

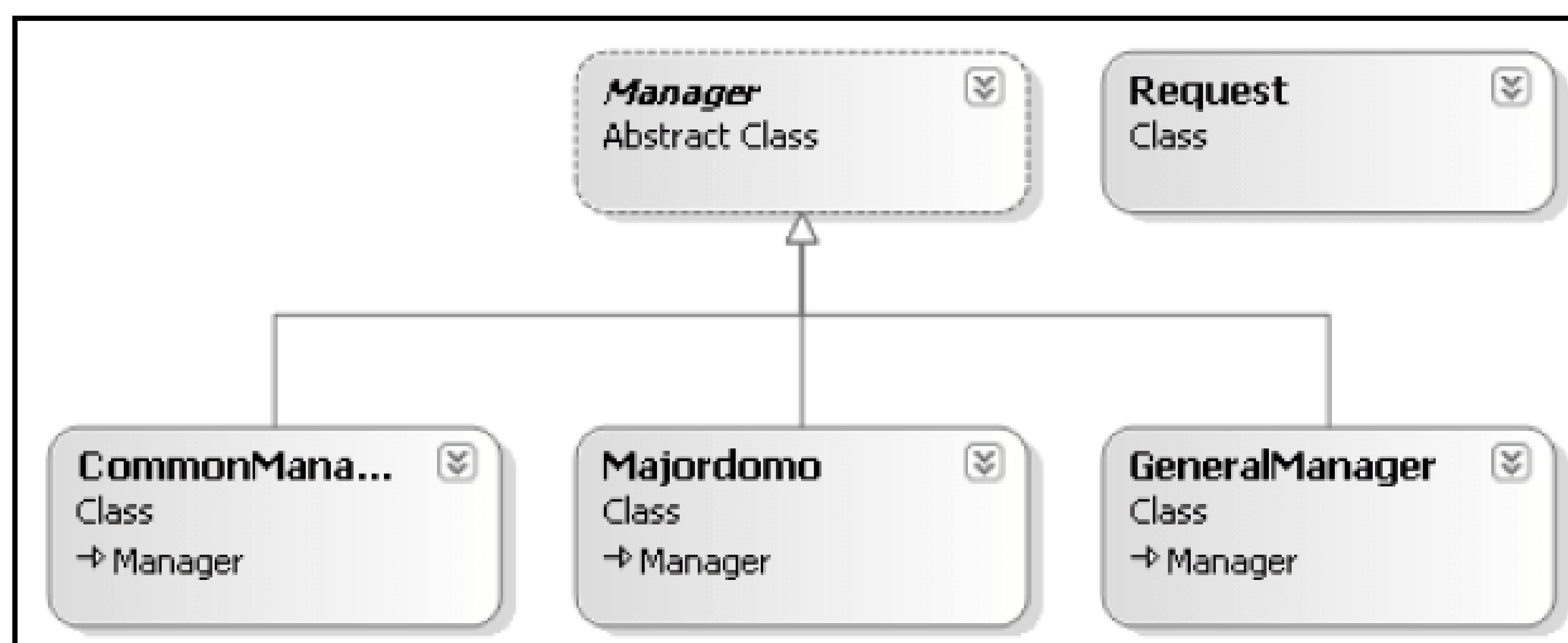


图 21.16　类关系设计图

设计过程

（1）定义申请类 Request，包含申请类别（如请假）、申请内容和申请数量（如请假天数）3 个属性。代码如下：

```
class Request                                   //申请
{
      public String RequestType ;               //申请类别
      public String RequestContent;             //申请内容
      public int RequestNumber;                 //申请数量
      public Request(String requestType, String requestContent, int requestNumber)
      {
            this.RequestType = requestType;
            this.RequestContent = requestContent;
            this.RequestNumber = requestNumber;
       }
}
```

（2）定义管理者抽象类，该类主要实现设置管理者的上级和处理请求。代码如下：

```
abstract class Manager                                //管理者抽象类
{
      protected String name;
      protected Manager superior;
      public Manager(String name)
      { this.name = name; }
      public void SetSuperior(Manager superior)       //设置管理者的上级
      {
            this.superior = superior;
      }
      abstract public void RequestApplications(Request request);
}
```

（3）定义部门经理类 CommonManager，继承 Manager 类，并实现 RequestApplications()方法。代码如下：

```
class CommonManager extends Manager                                          //部门经理
{
      public CommonManager(String name)
      {
         super(name);
      }
      public    void RequestApplications(Request request)
      {
            if (request.RequestType == "请假" && request.RequestNumber <= 1)     //部门经理有权准许下属 1 天内的假期
            {
                  System.out.println( name +":"+ request.RequestContent+ "  数量"+ request.RequestNumber+"  被批准");
            }
            else
            {
                  if (superior != null)
                  {
                        superior.RequestApplications(request);                  //其余的申请需转到上级
                  }
            }
       }
}
```

（4）定义总监类 Majordomo，继承 Manager 类，并实现 RequestApplications()方法。代码如下：

```
class Majordomo extends Manager                                      //总监
{
    public Majordomo(String name)
    {
      super(name);
    }
    public   void RequestApplications(Request request)
    {
        if (request.RequestType == "请假" && request.RequestNumber <= 3)          //总监有权准许下属 3 天内的假期
        {
           System.out.println( name +":"+ request.RequestContent+ "  数量"+ request.RequestNumber+"  被批准");
        }
        else
        {
          if (superior != null)
          {
              superior.RequestApplications(request);                  //其余的申请需转到上级
          }
        }

    }
}
```

（5）定义总经理类 GeneralManager，继承 Manager 类，并实现 RequestApplications()方法。代码如下：

```
class GeneralManager extends Manager                                 //总经理
{
    public GeneralManager(String name)
    {
      super(name);
    }
    public void RequestApplications(Request request)
    {
        if (request.RequestType == "请假")                            //总经理有权准许下属任意天的假期
        {
           System.out.println( name +":"+ request.RequestContent+ "   数量"+ request.RequestNumber+"   被批准");
        }
        else if (request.RequestType == "加薪" && request.RequestNumber <= 1000)   //加薪在 1000 以内，经理可批准
        {
           System.out.println( name +":"+ request.RequestContent+ "   数量"+ request.RequestNumber+"   被批准");
        }
        else                                                          //要求太高，不能批准
        {
           System.out.println( name +":"+ request.RequestContent+ "   数量"+ request.RequestNumber+"   被批准");
        }
    }
}
```

（6）在 main()方法中，分别创建部门经理、总监和总经理类的实例，然后设置部门经理的上级是总监，总监的上级是总经理，最后分别申请 1 天、3 天和 7 天假。代码如下：

```
class Program
{
   public static void main(String[] args)
    {
        CommonManager com = new CommonManager("房大伟");               //部门经理类
        Majordomo maj = new Majordomo("宋坤");                         //总监类
        GeneralManager gen = new GeneralManager("赛奎春");             //总经理类
        com.SetSuperior(maj);                                        //设置部门经理的上级是总监，可根据实际情况更改设置
        maj.SetSuperior(gen);                                        //设置总监的上级是总经理
        Request req1 = new Request("请假", "云峰要请假", 1);            //申请 1 天假
        com.RequestApplications(req1);                               //上级领导处理
        Request req2 = new Request("请假", "云峰要请假", 3);
        com.RequestApplications(req2);
        Request req3 = new Request("请假", "云峰要请假", 7);
        com.RequestApplications(req3);
    }
}
```

秘笈心法

心法领悟 535：责任链模式的优点和缺点。

- ❑ 降低耦合度：该模式使得一个对象无须知道是其他哪一个对象处理其请求，仅需知道该请求会被“正确”地处理。接收者和发送者都没有对方的明确信息，且链中的对象不需知道链的结构。结果是：责任链可简化对象的相互连接。它们仅需保持一个指向其后继者的引用，而不需保持所有的候选接受者的引用。
- ❑ 增强了给对象指派责任的灵活性：当在对象中分派责任时，责任链提供了更大的灵活性。用户可以将这种机制与静态的特例化处理对象的继承机制结合起来使用。
- ❑ 不保证被接受：既然一个请求没有明确的接收者，那么就不能保证它一定会被处理，可能一直到链的末端都得不到处理。另外，一个请求也可能因该链没有被正确配置而得不到处理。

实例 536	享元模式 光盘位置：光盘\MR\21\536	初级 实用指数：★★★★

实例说明

什么是享元（Flyweight）模式？笔者先通过一个实际生活中的例子进行介绍。

例如，项目经理要求创建 3 个产品展示网站、3 个博客网站，共需要 6 个网站类的实例。其实这 6 个网站的代码都一样，如果网站增多，实例也就随之增多，这对服务器的资源会造成严重的浪费，如图 21.17 所示。

如果将这 6 个网站整合到一个网站中，共享其相关的代码和数据，那么对于硬盘、内存、CPU、数据库空间等服务器资源都可以达成共享；更重要的是对于代码，由于只有一个实例，所以维护和扩展都更加容易。

这就需要用到享元模式。享元模式指运用共享技术有效地支持大量细粒度的对象。本实例将应用享元模式创建 3 个产品展示网站、3 个博客网站，运行结果如图 21.18 所示。

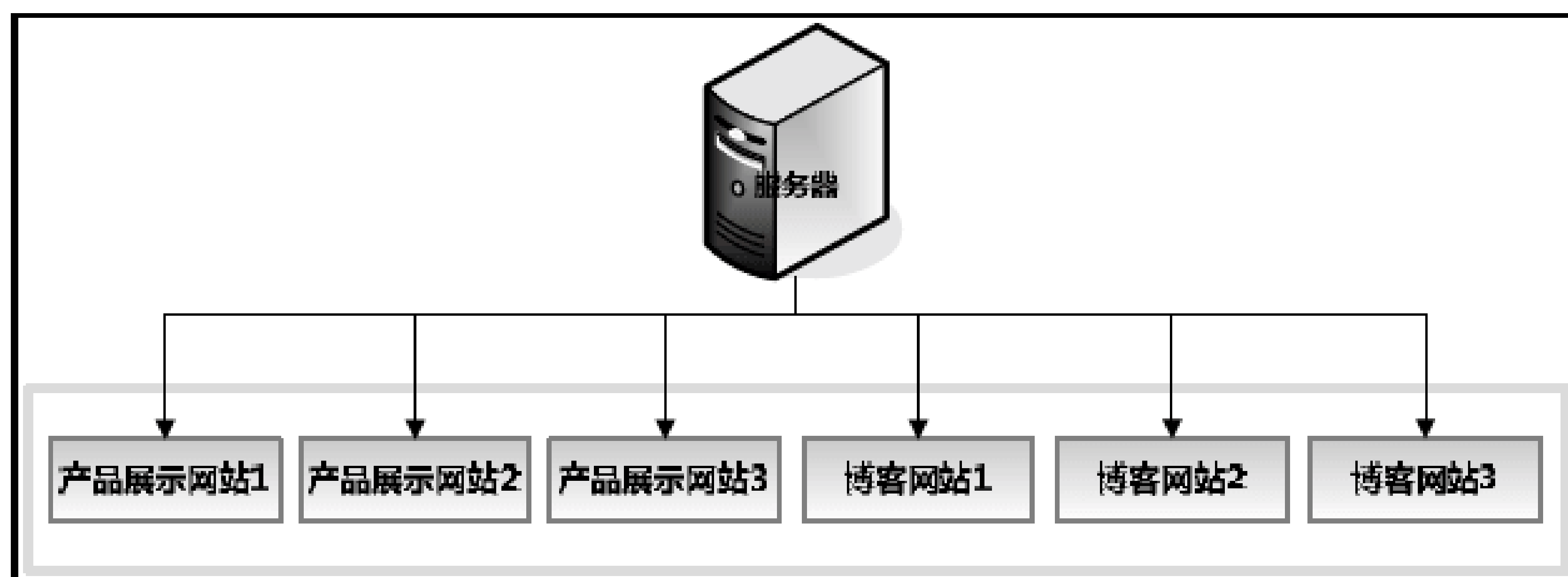


图 21.17　网站类的实例结构图



图 21.18　享元模式运行结果

提示： 如图 21.18 所示，虽然创建了 6 个网站，但下方显示的网站分类总数为 2，即 3 个产品展示网站共享一个实例，3 个博客网站共享另一个实例。

下面详细介绍享元模式的适用性和使用效果。

（1）适用性

享元模式的有效性很大程度上取决于如何使用以及在何处使用。当以下情况都成立时使用享元模式：

- ❑ 一个应用程序使用了大量的对象。
- ❑ 完全由于使用大量的对象，造成很大的存储开销。
- ❑ 对象的大多数状态都可变为外部状态。
- ❑ 如果删除对象的外部状态，那么可以用相对较少的共享对象取代很多组对象。
- ❑ 应用程序不依赖于对象标识。由于享元对象可以被共享，对于概念上明显有别的对象，标识测试将返回真值。

（2）使用效果

使用享元模式时，传输、查找或计算外部状态都会产生运行时的开销，尤其当享元被存储为内部状态时。

然而，空间上的节省抵消了这些开销。共享的享元越多，空间节省也就越大。

节约存储空间由以下几个因素决定。

- 因为共享，实例总数减少的数目。
- 对象内部状态的平均数目。
- 外部状态是计算的还是存储的。

共享的享元越多，节约的存储空间越大。也就是说，节约量随着共享状态的增多而增大。当对象使用大量的内部及外部状态，并且外部状态是计算出来的而非存储时，节约量将达到最大。因此可以用两种方法来节约存储空间：用共享减少内部状态的消耗；用计算时间换取对外部状态的存储。

（1）定义网站抽象类 WebSite，并声明 Use()方法。代码如下：

```
abstract class WebSite                                  //网站抽象类
{
    public abstract void Use();
}
```

（2）定义具体网站类 ConcreteWebSite，继承 WebSite 类。代码如下：

```
class ConcreteWebSite extends WebSite                   //具体网站
{
    private String name = "";
    public ConcreteWebSite(String name)
    {
        this.name = name;
    }
    public void Use()
    {
        System.out.println("网站分类：" + name);
    }
}
```

（3）定义网站工厂类，主要实现获得网站分类和统计网站分类总数。代码如下：

```
class WebSiteFactory                                    //网站工厂类
{
    private Hashtable flyweights = new Hashtable();
    public WebSite GetWebSiteCategory(String key)   //获得网站分类
    {
        if (!flyweights.contains(key))
        {
            flyweights.put(key, new ConcreteWebSite(key));
        }
        return (WebSite) flyweights.get(key) ;
    }

    public int GetWebSiteCount()                        //获得网站分类总数
    {
        return flyweights.size();
    }
}
```

（4）在 main()方法中，创建 3 个产品展示网站、3 个博客网站，并显示这些网站及网站分类总数。代码如下：

```
class Program
{
   public static void main(String[] args)
    {
        WebSiteFactory factory = new WebSiteFactory();
        WebSite site1 = factory.GetWebSiteCategory("产品展示");
        site1.Use();
        WebSite site2 = factory.GetWebSiteCategory("产品展示");
        site2.Use();
        WebSite site3 = factory.GetWebSiteCategory("产品展示");
        site3.Use();
        WebSite site4 = factory.GetWebSiteCategory("博客");
        site4.Use();
        WebSite site5 = factory.GetWebSiteCategory("博客");
```

```
            site5.Use();
            WebSite site6 = factory.GetWebSiteCategory("博客");
            site6.Use();
            System.out.println("购物网站分类总数为" + factory.GetWebSiteCount());
        }
}
```

秘笈心法

心法领悟 536：享元模式和组合模式的混合使用。

享元模式经常与组合模式结合起来表示一个层次式结构。组合模式实现层次的部分与整体的关系，享元模式实现层次中节点的共享。

实例 537　代理模式

光盘位置：光盘\MR\21\537

初级

实用指数：★★★★

实例说明

当程序人员需要将一个复杂的对象或创建时比较花费时间的对象表示成一个简单的对象时，可以使用代理（Proxy 或 Surrogate）模式。该模式可为其他对象提供一种代理以控制对该对象的访问。本实例通过代理模式实现数学计算中的加、减、乘、除计算，运行结果如图 21.19 所示。

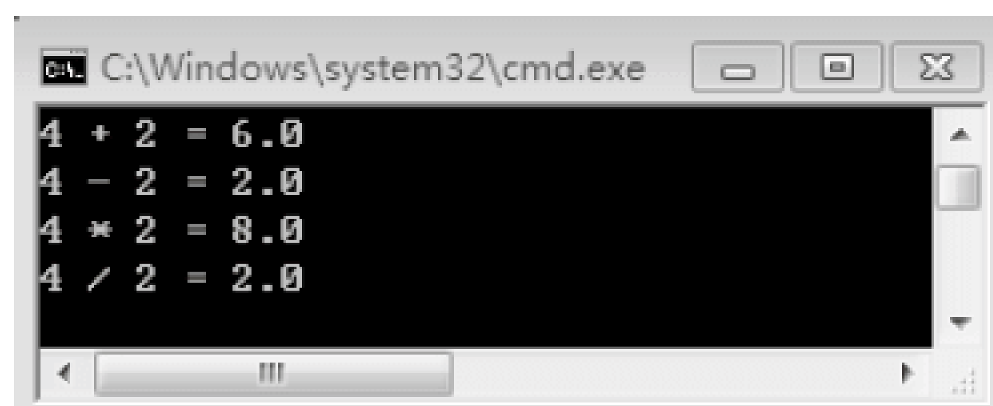


图 21.19　代理模式

关键技术

所谓代理，就是一个人或者一个机构代表另一个人或者另一个机构采取行动（或代替工作）。

下面通过一个生活中的“托儿所”例子来理解代理模式。

滕女士每天 8:00 将 3 岁的儿子送到托儿所，下班时再将孩子接回家。其间孩子由托儿所负责照看。

由此可见，上例中滕女士采用的就是代理模式。另外，Windows 中也有一个典型的例子，即快捷方式，快捷方式是它所引用的程序的一个代理。

以下情况需要使用代理模式。

- 一幅很大的图像，载入的时间很长。
- 一个需要很长时间才能完成的计算结果，并且需要在其计算过程中显示中间结果。
- 一个存在于远程计算机上的对象，想通过网络将其载入，则需要很长时间，特别是在网络传输高峰期。
- 一个对象只有有限的访问权限，代理模式可以验证用户的权限。

（1）定义数学计算的接口 IMath，并声明实现加、减、乘、除的方法。代码如下：

```
public interface IMath                    //数学计算的接口
{
      double Add(double x, double y);
      double Sub(double x, double y);
      double Mul(double x, double y);
      double Div(double x, double y);
}
```

（2）定义数学计算的具体类 Math，实现 IMath 接口。代码如下：

```
public class Math implements IMath{
      public double Add(double x, double y){return x + y;}
      public double Sub(double x, double y){return x - y;}
      public double Mul(double x, double y){return x * y;}
      public double Div(double x, double y){return x / y;}
}
```

（3）定义数学计算的代理类 MathProxy，实现 IMath 接口，并代理 IMath 接口的加、减、乘、除计算。代码如下：

```
class MathProxy implements IMath                //数学计算代理对象
{
    Math math;
    public MathProxy()
    {
        math = new Math();
    }

    public double Add(double x, double y)       //加法计算
    {
        return math.Add(x, y);
    }
    public double Sub(double x, double y)       //减法计算
    {
        return math.Sub(x, y);
    }
    public double Mul(double x, double y)       //乘法计算
    {
        return math.Mul(x, y);
    }
    public double Div(double x, double y)       //除法计算
    {
        return math.Div(x, y);
    }
}
```

（4）在 main()方法中，创建数学计算代理类的实例，并执行代理的加、减、乘、除计算。代码如下：

```
class Program
{
    public static void main(String[] args)
    {
        MathProxy p = new MathProxy();              //创建数学计算代理对象
        //执行加减乘除方法
        System.out.println("4 + 2 = " + p.Add(4, 2));
        System.out.println("4 - 2 = " + p.Sub(4, 2));
        System.out.println("4 * 2 = " + p.Mul(4, 2));
        System.out.println("4 / 2 = " + p.Div(4, 2));
    }
}
```

秘笈心法

心法领悟 537：适配器模式与代理模式的比较。

适配器模式和代理模式都是在对象外围构建了一个辅助层，但是适配器模式是为对象提供一个不同的接口，而代理模式为对象提供的是相同的接口，该接口可以推迟处理过程或数据转换工作。

21.3　构造型模式

实例 538	装饰模式 光盘位置：光盘\MR\21\538	初级 实用指数：★★★☆

实例说明

装饰模式又名包装模式，以对客户端透明的方式扩展对象的功能，是继承关系的一种替代方案。以下情况需要使用装饰模式。

- 需要扩展一个类的功能，或给一个类增加附加责任。

- 需要动态地给一个对象增加功能，这些功能可以再动态地撤销。
- 需要增加由一些基本功能的排列组合而产生的非常大的功能，从而使继承关系变得不现实。

本例通过装饰模式实现明日公司既可以开发计算机图书，也可以开发计算机软件。实例运行结果如图 21.20 所示。

关键技术

装饰模式是利用 SetComponent()（本例中是 SetWork()）方法来对对象进行包装的，这样每个装饰对象的实现就和如何使用这个对象分离了，每个装饰对象只关心自己的功能，不需要关心如何被添加到对象链当中。本实例中类的关系如图 21.21 所示。

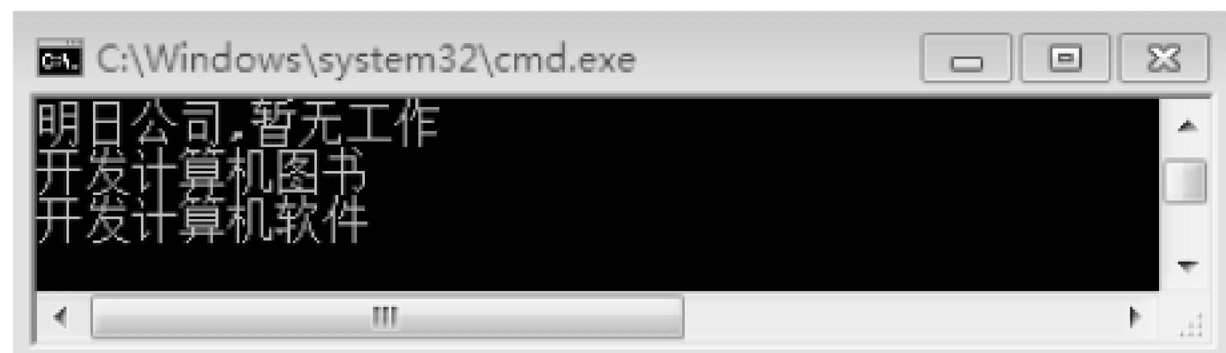


图 21.20　装饰模式

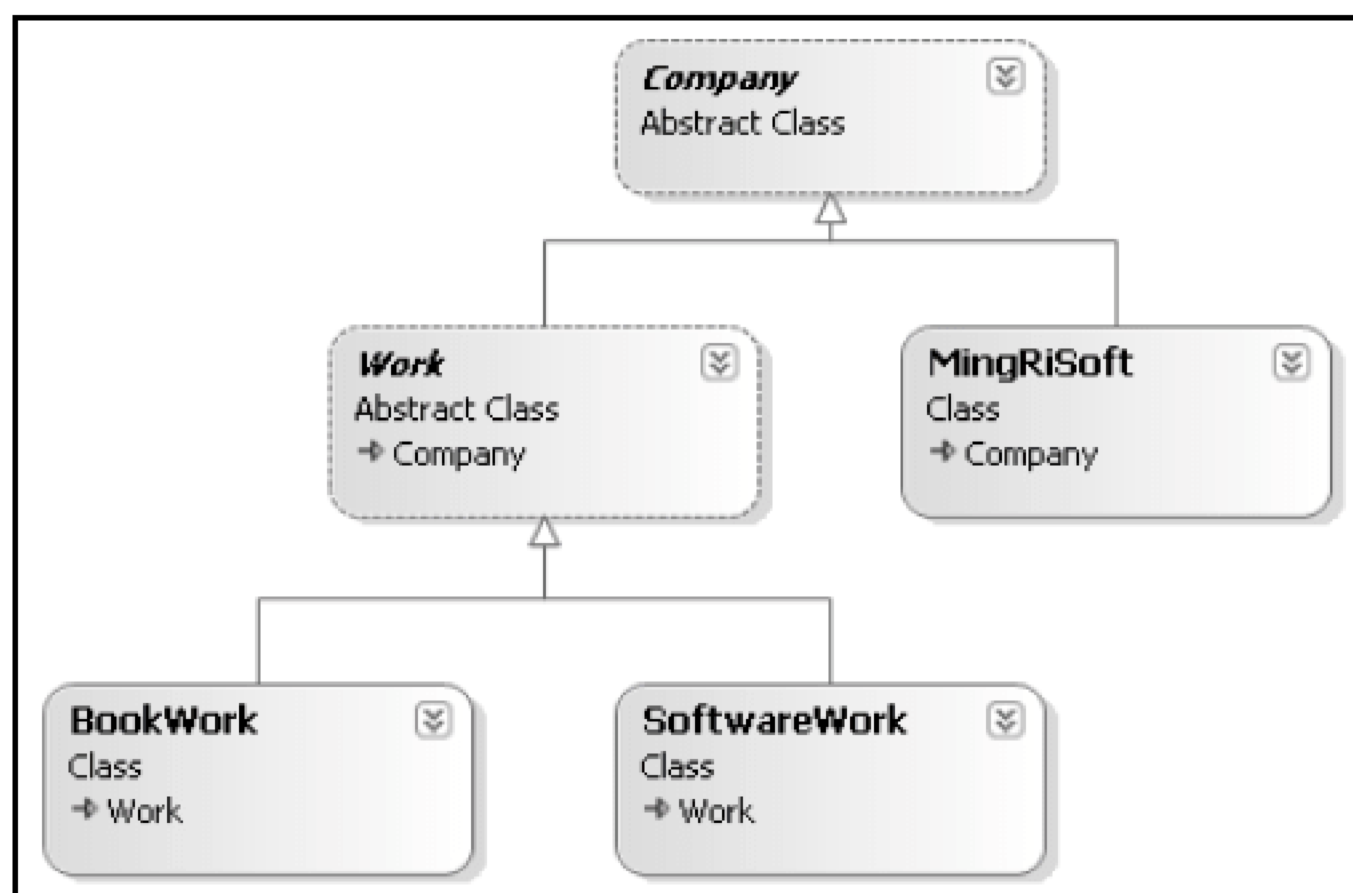


图 21.21　类关系设计图

说明： 在图 21.21 中，Company 是公司抽象类；Work 是工作抽象类；BookWork 是开发图书工作类；SoftwareWork 是开发软件工作类；MingRiSoft 是明日公司类。

设计过程

（1）定义公司抽象类 Company，并声明一个抽象方法 DoWork()。代码如下：

```
abstract class Company                          //公司抽象类
{
    public abstract void DoWork();
}
```

（2）定义明日公司类 MingRiSoft，继承 Company 类，并实现 DoWork()方法。代码如下：

```
class MingRiSoft extends Company                //明日公司
{
    public void DoWork()
    {
        System.out.println("明日公司,暂无工作");
    }
}
```

（3）定义工作抽象类 Work，继承 Company 类，并实现 DoWork()方法。代码如下：

```
abstract class Work extends Company             //工作抽象类
{
    protected Company component;
    public void SetWork(Company component)      //设置工作
    {
        this.component = component;
    }
    public void DoWork()                        //做工作
    {
        if (component != null)
        {
            component.DoWork();
        }
    }
}
```

技巧：上述代码中，Work 类不仅继承了 Company 类（实现 is a 的关系），而且还聚合了 Company 类（实现 has a 的关系）。

（4）定义开发图书工作类 BookWork，继承 Work 类，并实现 DoWork()方法。代码如下：

```
class BookWork extends Work              //开发图书工作
{
    public void DoWork()
    {
        super.DoWork();
        System.out.println("开发计算机图书");
    }
}
```

（5）定义开发软件工作类 SoftwareWork，继承 Work 类，并实现 DoWork()方法。代码如下：

```
class SoftwareWork extends Work          //开发软件工作
{
    public void DoWork()
    {
        super.DoWork();
        System.out.println("开发计算机软件");
    }
}
```

（6）在 main()方法中，分别创建明日公司类、开发图书工作类、开发软件工作类这 3 个类的实例，用开发图书工作对象组装明日公司对象，用开发软件对象组装开发图书对象。代码如下：

```
class Program
{
    public static void main(String[] args)
    {
        Company mr = new MingRiSoft();      //创建明日公司
        Work book = new BookWork();         //创建开发图书工作
        Work soft = new SoftwareWork();     //创建开发软件工作
        book.SetWork(mr);                   //指定明日公司可以开发图书
        soft.SetWork(book);                 //指定明日公司可以开发软件
        soft.DoWork();                      //做这些工作
    }
}
```

秘笈心法

心法领悟 538：使用装饰模式的优缺点。

装饰模式提供了一种给一个类添加职责的方式，比使用继承更加灵活，因为装饰模式将职责加到类的指定实例中；允许用户定制一个类，而无须在继承层次结构中创建高层次类。

（1）装饰模式的优点

装饰模式与继承关系的目的都是提供扩展对象的功能，但是装饰模式可以提供比继承更多的灵活性。

通过使用不同的具体装饰类以及这些装饰类的排列组合，程序设计者可以创造出很多不同行为的组合。

与继承相比，装饰模式具有更加灵活、机动的特性。

（2）装饰模式的缺点

使用装饰模式，可以比使用继承关系需要较少数目的类。使用较少的类，可使设计比较易于进行。但是，在另一方面，使用装饰模式会产生比使用继承关系更多的对象。更多的对象会使查错变得困难，特别是这些对象看上去都很相像。

实例 539	工厂方法模式 光盘位置：光盘\MR\21\539	初级 实用指数：★★★☆

实例说明

工厂方法（Factory Method）模式是类的创建模式，主要实现的是定义一个创建产品对象的工厂接口，将实

际创建工作推迟到子类中。

工厂方法模式是抽象工厂模式的进一步抽象和推广。由于使用了多态性，工厂方法模式保持了抽象工厂模式的优点，而且克服了其缺点。

在工厂方法模式中，核心的工厂类不再负责所有产品的创建，而是将具体创建工作交给子类完成。这个核心类仅仅负责给出具体工厂必须实现的接口，而不接触哪一个产品类被实例化这种细节。这使得工厂方法模式允许系统在不修改工厂角色的情况下引进新产品。

在工厂方法模式中，工厂类与产品类往往具有平行的等级结构且一一对应。

抽象工厂与工厂方法的区别如下。

工厂方法模式与抽象工厂模式在结构上的差异不是很明显，工厂方法类的核心是一个抽象工厂类，而抽象工厂模式把核心放在一个具体类上。

工厂方法模式的一个别名为多态性工厂模式，因为具体工厂类都有共同的接口，或者共同的抽象父类，当系统扩展需要添加新的产品对象时，仅仅需要添加一个具体对象以及一个具体工厂对象，原有工厂对象不需要进行任何修改，也不需要修改客户端，符合“开放－封闭”原则。而抽象工厂模式在添加新产品对象后不得不修改工厂方法，扩展性不好。工厂方法模式退化后可以演变成抽象工厂模式。

本实例中实现生产灯具的功能时应用了工厂方法模式，运行结果如图 21.22 所示。

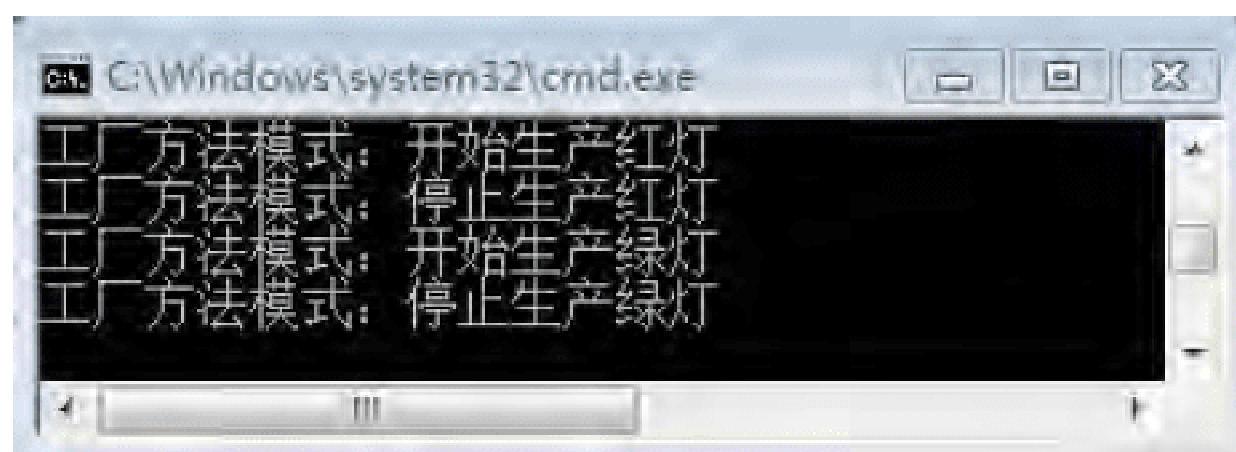


图 21.22　工厂方法模式

工厂方法模式中涉及的类的关系如图 21.23 所示。

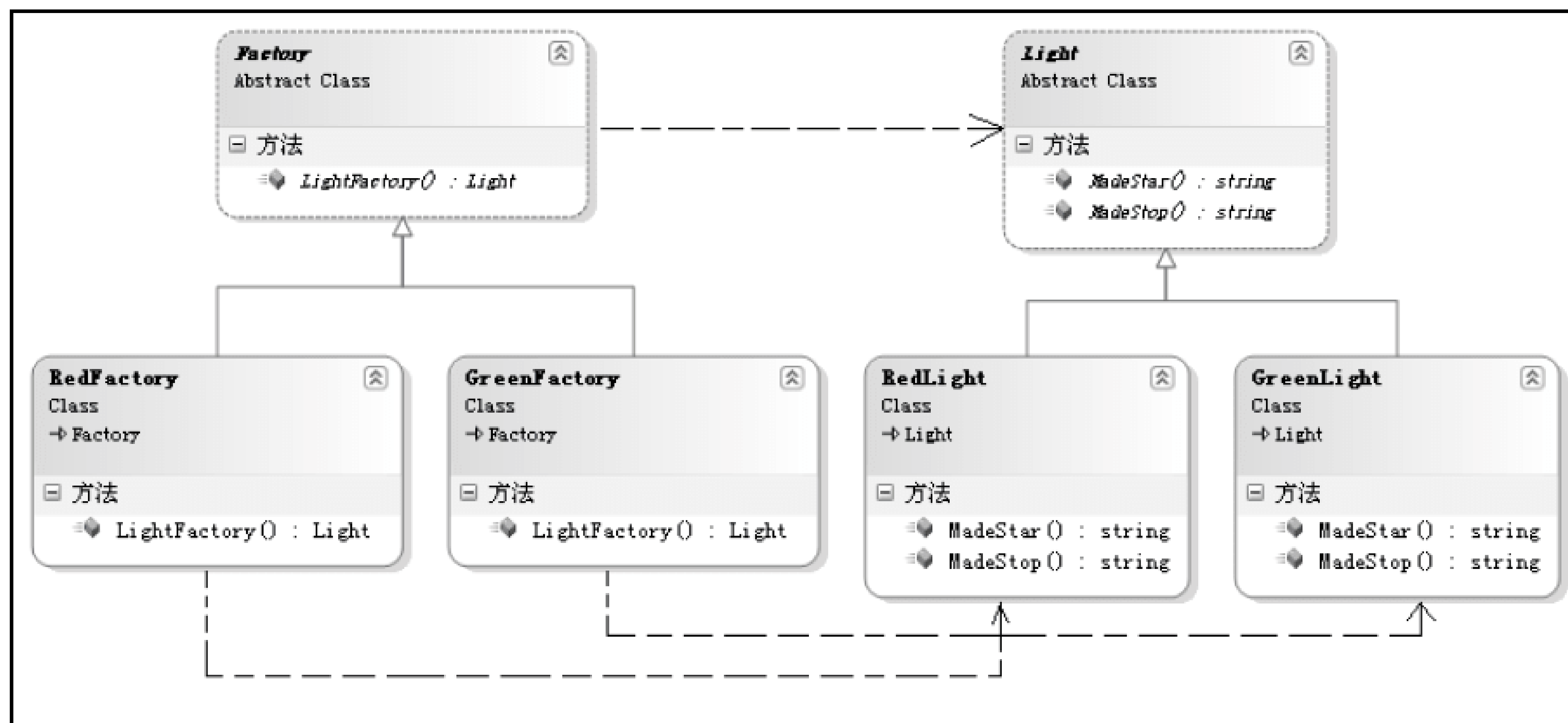


图 21.23　工厂方法模式类关系设计图

- 抽象工厂类（Factory）：是工厂方法模式的核心，与应用程序无关。任何在模式中创建的对象的工厂类必须实现这个接口。
- 具体工厂类（RedFactory 和 GreenFactory）：是实现抽象工厂接口的具体工厂类，包含与应用程序密切相关的逻辑，并且受到应用程序调用以创建产品对象。
- 抽象产品类（Light）：工厂方法模式所创建的对象的超类型，也就是产品对象的共同父类或共同拥有的接口。

- 具体产品类（RedLight 和 GreenLight）：这个角色实现了抽象产品角色所定义的接口。某具体产品由专门的具体工厂创建，它们之间往往一一对应。

（1）定义灯的抽象类 Light，并声明两个抽象方法。代码如下：

```
public abstract class Light
{
    public abstract string MadeStar();    //开发生产
    public abstract string MadeStop();    //停止生产
}
```

（2）定义红灯类 RedLight 并继承抽象类 Light，然后实现 MadeStar()和 MadeStop()抽象方法。代码如下：

```
public class RedLight : Light
{
    public override string MadeStar()
    {
        return "方法工厂模式：开始生产红灯";
    }
    public override string MadeStop()
    {
        return "方法工厂模式：停止生产红灯";
    }
}
```

（3）定义绿灯类 GreenLight 并继承抽象类 Light，然后实现 MadeStar()和 MadeStop()抽象方法。代码如下：

```
public class GreenLight : Light
{
    public override string MadeStar()
    {
        return "方法工厂模式：开始生产绿灯";
    }
    public override string MadeStop()
    {
        return "方法工厂模式：停止生产绿灯";
    }
}
```

（4）定义抽象工厂类 Factory 并声明抽象方法。Factory 类作为抽象工厂角色，是工厂方法模式的核心，任何在模式中创建的对象的工厂类必须实现这个接口。代码如下：

```
public abstract class Factory
{
    public abstract Light LightFactory();
}
```

（5）定义生产红灯的工厂类 RedFactory，这是实现抽象工厂接口的具体工厂类，包含与应用程序密切相关的逻辑，并且受到应用程序调用以创建产品对象。代码如下：

```
public class RedFactory:Factory
{
    public override Light LightFactory()
    {
        return new RedLight();
    }
}
```

（6）定义生产绿灯的工厂类 GreenFactory，这是实现抽象工厂接口的具体工厂类，包含与应用程序密切相关的逻辑，并且受到应用程序调用以创建产品对象。代码如下：

```
public class GreenFactory : Factory
{
    public override Light LightFactory()
    {
        return new GreenLight();
    }
}
```

（7）在 main()方法中，分别创建红灯工厂对象、绿灯工厂对象，然后使用这两个工厂对象分别生产红灯和绿灯。代码如下：

```
class Program
{
    static void Main(string[] args)
    {    //分别创建红灯工厂对象、绿灯工厂对象
        Factory redFactory = new RedFactory();
        Factory greenFactory = new GreenFactory();
        //利用红灯工厂对象创建红灯
        Light redLight = redFactory.LightFactory();
        Console.WriteLine(redLight.MadeStar() +"\n"+ redLight.MadeStop());
        //利用绿灯工厂对象创建绿灯
        Light greenLight = greenFactory.LightFactory();
        Console.WriteLine(greenLight.MadeStar() +"\n"+ greenLight.MadeStop());
        Console.Read();
    }
}
```

秘笈心法

心法领悟 539：抽象工厂和工厂方法的区别。

抽象工厂模式的最大优点在于工厂类中包含了必要的逻辑判断，根据客户端的选择条件动态实例化相关的类，对客户端而言，除去了与具体产品的依赖。

工厂方法模式实现时，客户端需要决定实例化哪一个工厂来实现运算类，选择判断的问题还是存在的，也就是说，工厂方法把抽象工厂内部逻辑判断移到了客户端代码中进行。如果要添加新的功能，原本是修改工厂类，而现在则是修改客户端。



实例 540	抽象工厂模式 光盘位置：光盘\MR\21\540	初级 实用指数：★★★★

实例说明

在面向对象程序设计中，对程序开发人员而言，最常见的设计模式就是抽象工厂模式。抽象工厂模式根据提供给它的数据，返回几个类中的一个类的实例。通常返回的类都有一个公共父类和公共方法，但是每个方法实现的功能不同，并且根据不同的数据进行初始化。抽象工厂模式具有如下优缺点。

优点：工厂类含有必要的判断逻辑，可以决定在什么时候创建哪一个产品类的实例，客户端可以免除直接创建产品对象的责任，而仅仅“消费”产品。抽象工厂模式通过这种做法实现了对责任的分割。

缺点：当产品具有复杂的多层等级结构时，工厂类只有自身，不随其他结构变化而变化，这是该模式的最大缺点。因为工厂类集中了所有产品的创建逻辑，一旦不能正常工作，整个系统都要受到影响。同时，系统扩展困难，一旦添加新产品就不得不修改工厂逻辑，有可能造成工厂逻辑过于复杂。

本实例中实现生产灯具的功能时应用了抽象工厂模式，运行结果如图 21.24 所示。



图 21.24　抽象工厂模式

下面结合抽象工厂角色与结构图（如图 21.25 所示）了解一下面向对象思想。

- 工厂类角色 LightSimpleFactory：工厂类在客户端的直接控制下（Create 方法）创建产品对象。
- 抽象产品角色 Light：定义抽象工厂创建的对象的父类或各角色共同拥有的接口。可以是一个类、抽象

类或接口。

- 具体产品角色 RedRight（红灯）、GreenLight（绿灯）：定义工厂具体加工出的对象。

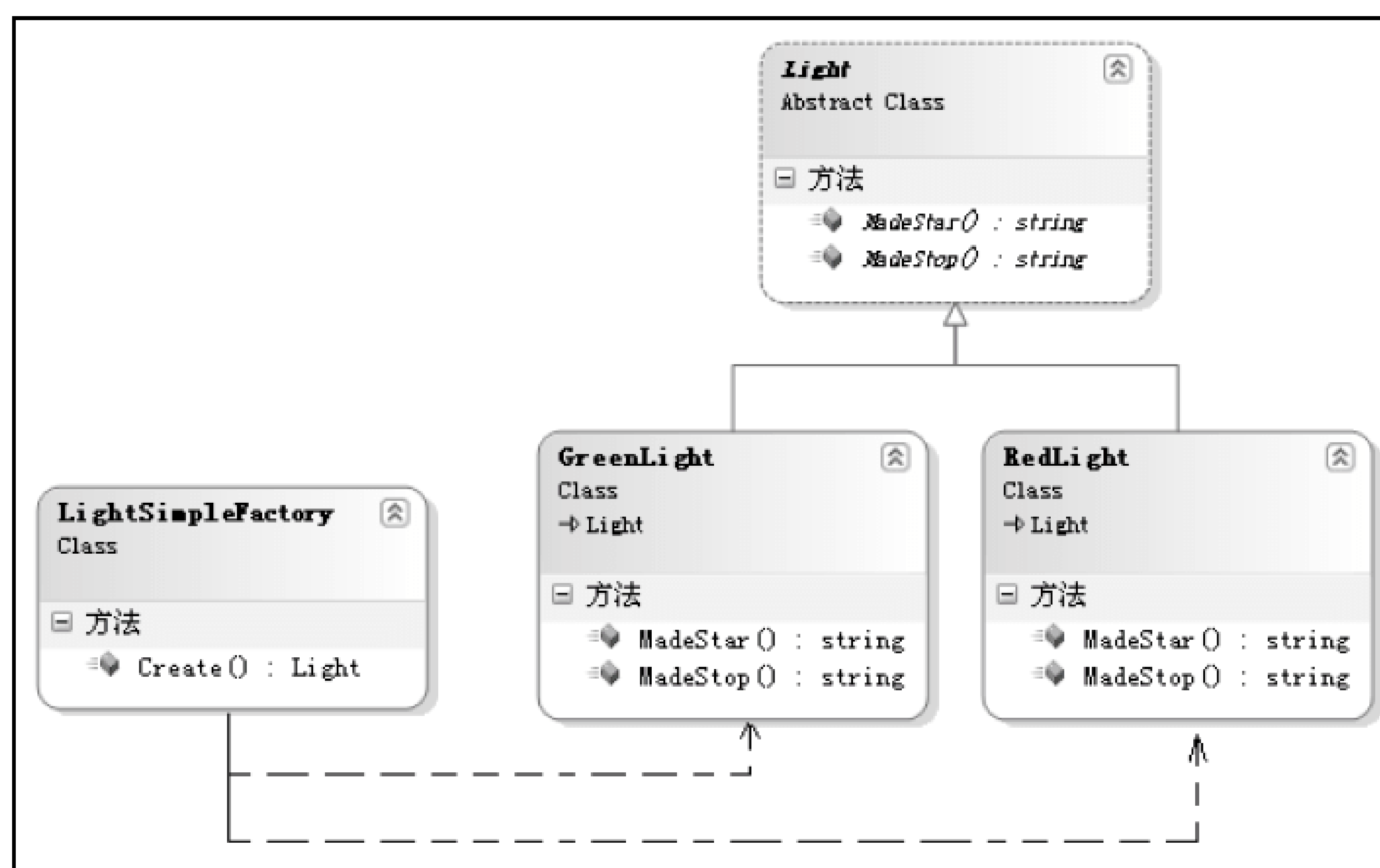


图 21.25　抽象工厂角色与结构图

（1）在 main()方法中调用工厂类 LightSimpleFactory 中的 Create()方法，创建产品对象。代码如下：

```
public class Program
{
    public static void main(String[] args)
    {
        LightSimpleFactory lsf = new LightSimpleFactory();
        Light l = lsf.Create("绿灯");
        l.TurnOn();
        l.TurnOff();
        System.out.println("----------------");
        l = lsf.Create("红灯");
        l.TurnOn();
        l.TurnOff();
    }
}
```

（2）定义抽象类 Light，并声明两个抽象方法。代码如下：

```
public abstract class Light          //定义抽象类
{
   public abstract void TurnOn();
   public abstract void TurnOff();
}
```

（3）定义类 RedLight，继承抽象类 Light，然后实现 TurnOn()和 TurnOff()方法。代码如下：

```
public class RedLight extends Light
{
    public void TurnOn()
    {
        System.out.println("开始生产红灯！");
    }
    public void TurnOff()
    {
        System.out.println("生产红灯结束！");
    }
}
```

（4）定义类 GreenLight，继承抽象类 Light，然后实现 TurnOn()和 TurnOff()方法。代码如下：

```
public class GreenLight extends Light
{
    public void TurnOn()
    {
```

```
            System.out.println("开始生产绿灯！");
        }
        public    void TurnOff()
        {
            System.out.println("生产绿灯结束！");
        }
}
```

（5）定义 LightSimpleFactory 类，用于实现工厂生产，通过 Create()方法生产红灯和绿灯。代码如下：

```
public class LightSimpleFactory
{
        public Light Create(String LightColor)
        {
            if (LightColor == "绿灯")
                return new GreenLight();
            else if (LightColor == "红灯")
                return new RedLight();
            else
                return null;
}
}
```

秘笈心法

心法领悟 540：在工厂模式中应用反射。

在本实例的 LightSimpleFactory 类的 Create()方法中，通过条件判断确定是创建红灯（RedLight）对象还是创建绿灯（GreenLight）对象，可以将这些信息写入配置文件中，根据配置文件的信息通过反射创建相应的实例对象。

实例 541	原型模式 光盘位置：光盘\MR\21\541	初级 实用指数：★★★☆

实例说明

什么是原型（Prototype）模式？笔者先通过一个实际生活中的例子进行介绍。例如，一位美女来到理发店，对发型设计师说：“请给我设计一下发型。”发型设计师问她：“那您想设计成什么样的发型？”美女看了看四周，指着旁边一位顾客说：“就设计一款和她一模一样的发型。”

从美女顾客的角度考虑，使用的是原型模式，如果不是这样，美女顾客将和发型设计师一起花时间讨论。

本实例中利用原型模式实现克隆颜色，运行结果如图 21.26 所示。

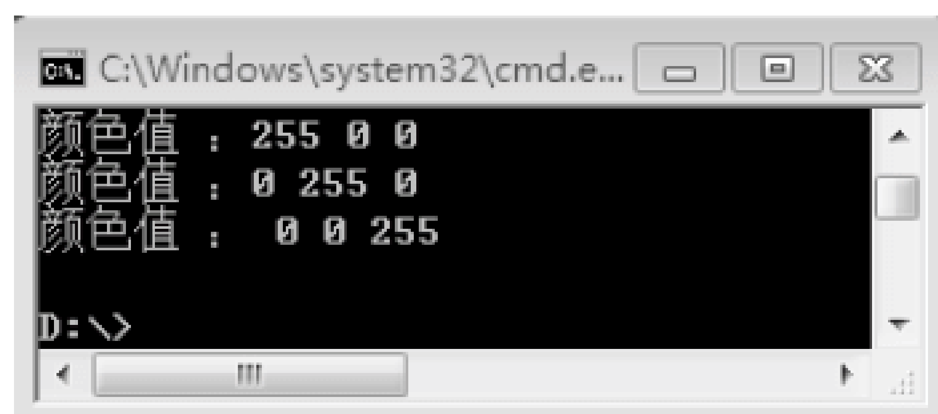


图 21.26　原型模式

关键技术

所谓原型模式，就是用原型对象指定所要创建的对象的种类，然后通过克隆这个原型对象的方法创建出更多的同类型对象。

原型模式的优点包括：

- ❑ 原型模式允许动态增加或减少产品类。由于创建产品类实例的方法是产品类内部具有的，因此增加新产品对整个结构没有影响。
- ❑ 原型模式提供了简化的创建结构。工厂方法模式常常需要有一个与产品类等级结构相同的等级结构，而原型模式则不需要。
- ❑ 原型模式具有给一个应用软件动态加载新功能的能力。由于原型模式的独立性较高，可以很容易地动态加载新功能而不影响旧系统。

产品类不必有任何事先确定的等级结构，因为原型模式适用于任何等级结构。

原型模式的缺点是每一个类必须配备一个克隆方法，而且采用这个克隆方法需要对类的功能进行周全的考虑。这对全新的类来说不是很难实现，但对现有的类进行改造时，并不很容易实现。

（1）定义一个抽象类，并声明一个抽象克隆方法 Clone()。代码如下：

```
public abstract class ColorRGBPrototype
{
    //定义抽象方法（克隆）
    public abstract   void Clone(String key,ColorRGB value) ;
}
```

（2）定义一个类 ColorRGB，该类主要用于实现克隆方法 Clone()。代码如下：

```
class ColorRGB extends ColorRGBPrototype
{
    private int red, green, blue;
    ColorRGBManager crm = new ColorRGBManager();
    public ColorRGB(int red, int green, int blue)                    //设置颜色值
    {
        this.red = red;
        this.green = green;
        this.blue = blue;
    }
    public ColorRGB() {}
public   void Clone(String key,ColorRGB colorRGB)
    {
      crm.manager(key, colorRGB);
    }
    public void Displayr()                                            //输出克隆后的颜色值
    {
        System.out.println("颜色值 ： " + ((ColorRGB)crm.colors.get("red")).red
                                +" "+   ((ColorRGB)crm.colors.get("red")).green
                                + " "+((ColorRGB)crm.colors.get("red")).blue);
    }
    public void Displayg()                                            //输出克隆后的颜色值
    {
        System.out.println("颜色值 ： " + ((ColorRGB)crm.colors.get("green")).red
                                +" "+   ((ColorRGB)crm.colors.get("green")).green
                                + " "+((ColorRGB)crm.colors.get("green")).blue);
    }
    public void Displayb()                                            //输出克隆后的颜色值
    {
        System.out.println("颜色值 ：  " + ((ColorRGB)crm.colors.get("blue")).red
                                +" "+((ColorRGB)crm.colors.get("blue")).green
                                + " "+((ColorRGB)crm.colors.get("blue")).blue);
    }
    public void Remove(){
      crm.remanager();
    }
}
```

（3）定义一个类 ColorRGBManager，该类用于设置或读取将要克隆对象的名称（起到一个原型管理器的作用）。代码如下：

```
class ColorRGBManager
{       String name;
    Hashtable colors = new Hashtable();
    public void manager(String key,ColorRGB colorRGB)
    {
      colors.put(key, colorRGB);
    }
    public void remanager(){
      colors.clear();
    }
}
```

（4）在 main()方法中利用原型设计模式实现克隆颜色，代码如下：

```
public class Program
{
    public static void main(String[] args)
    {
      ColorRGBManager redmanager = new ColorRGBManager() ;
      ColorRGB crgb = new ColorRGB() ;//创建颜色管理对象
        //开始克隆
        String colorName = "red";
        crgb.Clone(colorName, new ColorRGB(255, 0, 0));
        crgb.Displayr();
        colorName = "green";
        crgb.Clone(colorName, new ColorRGB(0, 255, 0));
        crgb.Displayg();
        colorName = "blue";
        crgb.Clone(colorName, new ColorRGB(0, 0, 255));
        crgb.Displayb();
    }
}
```

秘笈心法

心法领悟 541：原型模式的工作原理。

原型模式允许一个对象再创建另外一个可定制的对象，无须知道任何关于创建可定制对象的细节。其工作原理是：通过将一个原型对象传给要发动创建的对象，这个要发动创建的对象通过请求原型对象来复制自身，然后实施创建。

实例 542　备忘录模式

光盘位置：光盘\MR\21\542

初级

实用指数：★★★☆

实例说明

什么是备忘录（Memento）模式？笔者先通过一个实际生活中的例子进行介绍。

例如，玩单机 RPG 游戏时，玩到一定阶段时就保存一下进度，当游戏任务执行失败时，可以读取最后一次保存的进度，重新开始玩。这里就应用到了备忘录模式。

本实例中应用备忘录模式实现对某一对象的状态进行保存和恢复，运行结果如图 21.27 所示。

应用备忘录模式之前必须掌握备忘录模式的概念及适用性，下面分别介绍。

（1）备忘录模式的概念

在不破坏封装性的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态。这样以后就可以将该对象恢复到原先保存的状态。本实例中类的关系如图 21.28 所示。

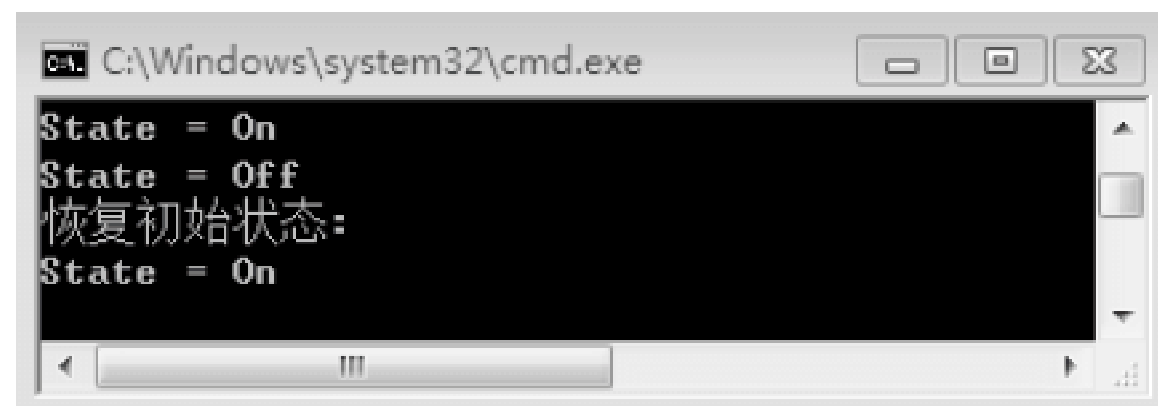


图 21.27　备忘录模式

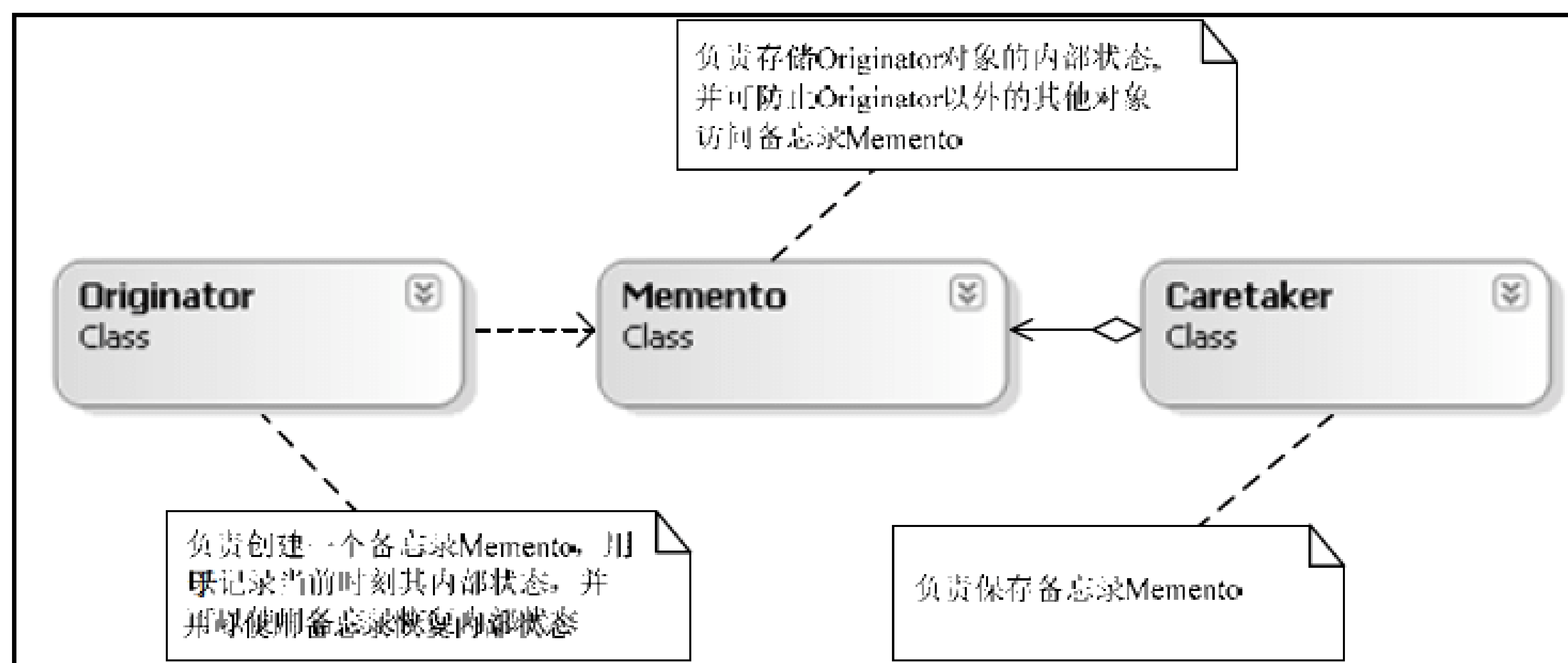


图 21.28　类关系设计图

（2）备忘录模式的适用性

必须保存一个对象在某一个时刻的（部分）状态，这样以后需要时它才能恢复到先前的状态。

如果使用接口让其他对象直接得到这些状态，将会暴露对象的实现细节并破坏对象的封装性。

■

（1）定义发起人类 Originator。代码如下：

```
class Originator                                    //发起人类
{
    String state;
    static Memento temp;
    public String State(String str)                 //需要保存的属性，可能有多个
    {
            System.out.println("State = " + str);
            return str;
    }
    public Memento CreateMemento(String state)      //创建备忘录，将当前需要保存的信息导入并实例化一个 Memento 对象
    {
            temp = new Memento(state);
            return temp;
    }
    public void SetMemento(Memento memento)         //恢复备忘录，将 Memento 导入并将相关数据恢复
    {
             System.out.println("恢复初始状态:");
             System.out.println("State = " + temp.state);

    }
}
```

（2）定义备忘录类 Memento。代码如下：

```
class Memento                                       //备忘录类
{
    String state;
    public Memento(String state)                    //构造函数，将相关数据导入
    {
        this.state = state;
    }
    public String State()                           //需要保存的数据属性，可以是多个
    {
        return state;
    }
}
```

（3）定义管理者类 Caretaker。代码如下：

```
class Caretaker                                     //管理者
{
    Memento memento;
    public Memento Memento()                        //得到或设置备忘录
    {
        return memento;
    }
}
```

（4）在 main()方法中，调用 Originator 类的实例对象的相关方法保存及恢复对象的状态。代码如下：

```
class Program
{
    public static void main(String[] args)
    {
        Originator o = new Originator();
        String str = o.State("On");                 //初始状态为 On
        Caretaker c = new Caretaker();
        c.memento = o.CreateMemento(str);           //保存状态时，由于有了很好的封装，可以隐藏 Originator 的实现细节
        o.State("Off");                             //Originator 改变了状态属性为 Off
        o.SetMemento(c.memento);                    //恢复初始状态
    }
}
```

秘笈心法

心法领悟 542：使用备忘录模式达到的效果。

- 保持封装边界。使用备忘录可以避免暴露一些只应由原发器管理却又必须存储在原发器之外的信息。该模式把可能很复杂的 Originator 内部信息对其他对象屏蔽起来，从而保持了封装边界。
- 简化了原发器。在其他保持封装性的设计中，Originator 负责保持客户请求过的内部状态版本。这就把所有存储管理的重任交给了 Originator。让客户管理它们请求的状态将会简化 Originator，并且使得客户工作结束时无须通知原发器。
- 使用备忘录可能代价很高。如果原发器在生成备忘录时必须复制并存储大量的信息，或者客户非常频繁地创建备忘录和恢复原发器状态，可能会导致非常大的开销。除非封装和恢复 Originator 状态的开销不大，否则该模式可能并不适用。
- 定义窄接口和宽接口，在一些语言中可能难以保证只有原发器可访问备忘录的状态。
- 维护备忘录的潜在代价，管理器负责删除它所维护的备忘录，但并不知道备忘录中有多少个状态，因此当存储备忘录时，一个本来很小的管理器可能会产生大量的存储开销。

21.4　行为型模式

实例 543	命令模式 光盘位置：光盘\MR\21\543	初级 实用指数：★★★★

实例说明

什么是命令模式？笔者先通过一个实际生活中的例子进行介绍。

例如，去餐馆吃饭，向服务员点了一盘烧茄子，服务员将此要求告诉给厨师。这个点菜流程可以应用命令模式来实现，服务员向厨师发出命令，厨师只需根据命令下达的顺序执行命令，即根据所下菜单做菜，而不用考虑菜是哪位顾客点的。本实例主要演示如何应用命令模式来实现点菜流程，运行结果如图 21.29 所示。

关键技术

应用命令模式之前必须掌握命令模式的概念及适用性，下面分别介绍。

（1）命令模式的概念

命令模式是将一个请求封装为一个对象，从而使用户可用不同的请求对客户进行参数化；对请求排队或记录请求日志，以及支持可撤销的操作。本实例中类的关系如图 21.30 所示。

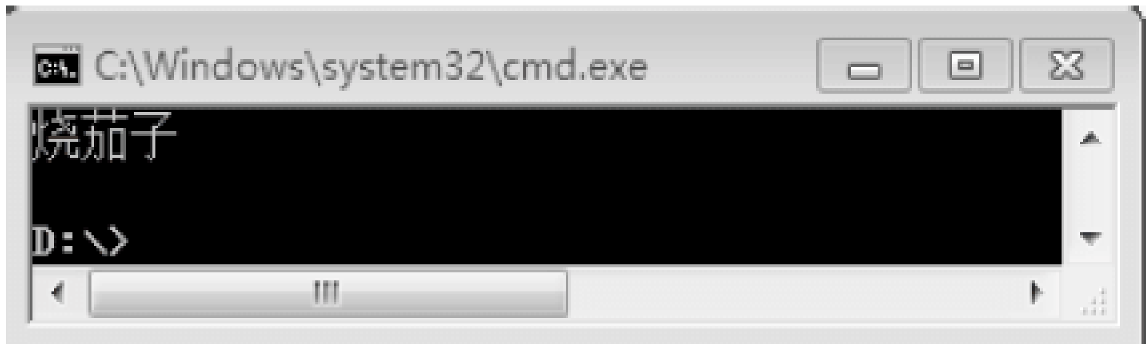


图 21.29　命令模式

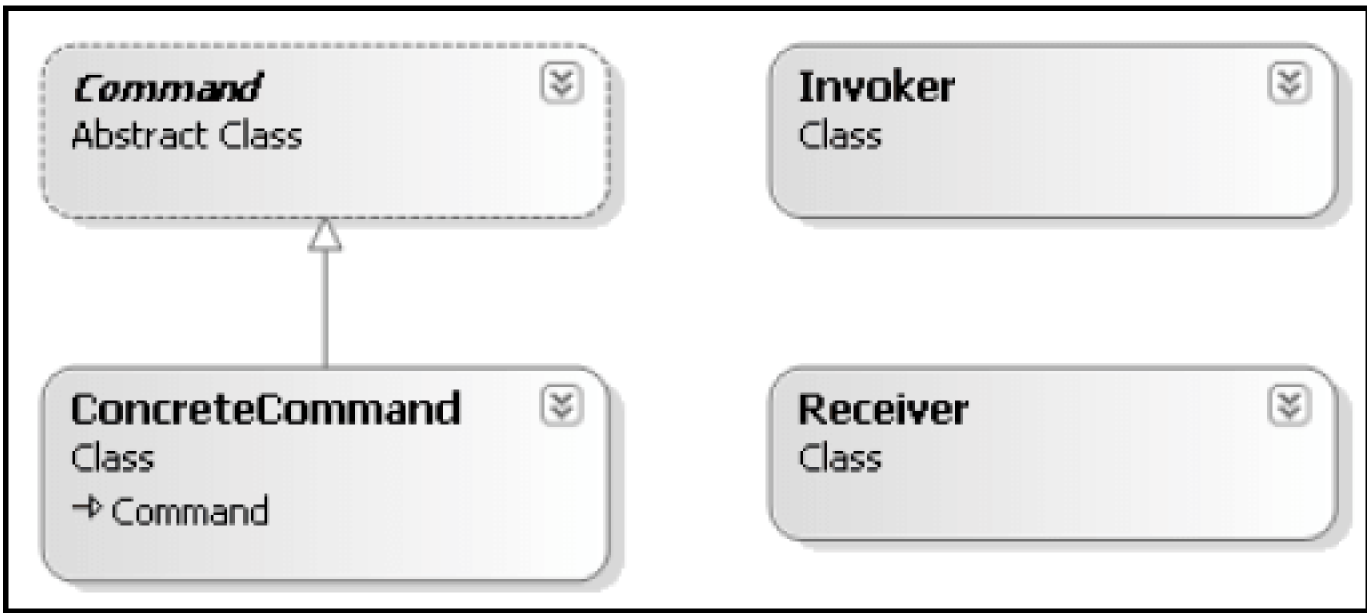


图 21.30　类关系设计图

说明：在图 21.30 中，Command 是命令抽象类；ConcreteCommand 是具体命令类；Receiver 是命令接受者类（做烧茄子的厨师）；Invoker 是命令执行者类（服务员）。

（2）命令模式的适用性

满足以下条件时可以使用 Command 模式。

- 在不同的时刻指定、排列和执行请求。一个 Command 对象可以有一个与初始请求无关的生存期。如果一个请求的接收者可用一种与地址空间无关的方式表达，那么就可以将负责该请求的命令对象传送给另一个不同的进程并在那里实现该请求。
- 支持取消操作。Command 的 Execute 操作可在实施操作前将状态存储起来，在取消操作时这个状态用来消除该操作的影响。Command 接口必须添加一个 UnExecute 操作，该操作用于取消上一次 Execute 调用的效果。执行的命令被存储在一个历史列表中，可通过向后和向前遍历这一列表并分别调用 UnExecute 和 Execute 来实现重复次数不限的“取消”和“重做”。
- 支持修改日志，这样当系统崩溃时，这些修改可以被重做一遍。在 Command 接口中添加装载操作和存储操作，可以针对变动保持一个一致的修改日志。从崩溃中恢复的过程包括从磁盘中重新读入记录下来的命令并用 Execute 操作重新执行它们。
- 用构建在原语操作上的高层操作构造一个系统，这样一种结构在支持事务的信息系统中很常见。一个事务封装了对数据的一组变动。Command 模式提供了对事务进行建模的方法。它有一个公共的接口，使得用户可以用同一种方式调用所有的事务。同时，使用该模式也易于添加新事务以扩展系统。

（1）定义命令抽象类 Command。代码如下：

```
abstract class Command                                    //命令抽象类
{
      protected Receiver receiver;
      public Command(Receiver receiver)
      {
            this.receiver = receiver;
      }
      public abstract void Execute();
}
```

（2）定义具体命令类 ConcreteCommand（做烧茄子）。代码如下：

```
class ConcreteCommand extends Command                    //具体命令
{
      public ConcreteCommand(Receiver receiver)          //构造函数
      {
         super(receiver);
      }
      public    void Execute()
      {
            receiver.Action();
      }
}
```

（3）定义命令接受者类 Receiver（做烧茄子的厨师）。代码如下：

```
class Receiver                                            //接受者
{
      public void Action()
      {
            System.out.println("烧茄子");
      }
}
```

（4）定义命令执行者类 Invoker（服务员）。代码如下：

```
class Invoker                                             //执行者
{
      private Command command;
      public void SetCommand(Command command)
      {
            this.command = command;
      }
      public void ExecuteCommand()
```

```
        {
            command.Execute();
        }
}
```

（5）在 main()方法中，分别创建接受者、命令和执行者类的实例，然后执行者设置并执行命令。代码如下：

```
class Program
{
    public static void main(String[] args)
    {
        Receiver receiver = new Receiver();                    //创建接受者
        Command command = new ConcreteCommand(receiver);   //创建命令
        Invoker invoker = new Invoker();                       //创建执行者
        //设置并执行命令
        invoker.SetCommand(command);
        invoker.ExecuteCommand();
     }
}
```

秘笈心法

心法领悟 543：命令模式实现的效果。

Command 模式将调用操作的对象与知道如何实现该操作的对象解耦。

Command 是头等的对象，可以像其他的对象一样被操纵和扩展。

可以将多个命令装配成一个复合命令。一般说来，复合命令是 Composite 模式的一个实例。

增加新的 Command 很容易，因为这无须改变已有的类。

实例 544　解释器模式

光盘位置：光盘\MR\21\544

初级

实用指数：★★★☆

实例说明

什么是解释器模式？可以通过一个简单的例子来说明。

例如，搜索匹配某一模式的字符串是一种很常见的问题。正则表达式是描述字符串模式的一种标准语言。与其为每一种模式都构造一个特定的算法，不如使用一种通用的搜索算法来解释执行一个正则表达式，该正则表达式定义了待匹配字符串的集合。

本实例中应用解释器模式实现用终端、非终端这两种解释器来解释特定的内容，运行结果如图 21.31 所示。

关键技术

应用解释器模式之前必须掌握解释器模式的概念及适用性，下面分别介绍。

（1）解释器模式的概念

解释器模式是给定一种语言，定义其文法的一种表示，并定义一个解释器，这个解释器使用该表示来解释语言中的句子。本实例中类的关系如图 21.32 所示。

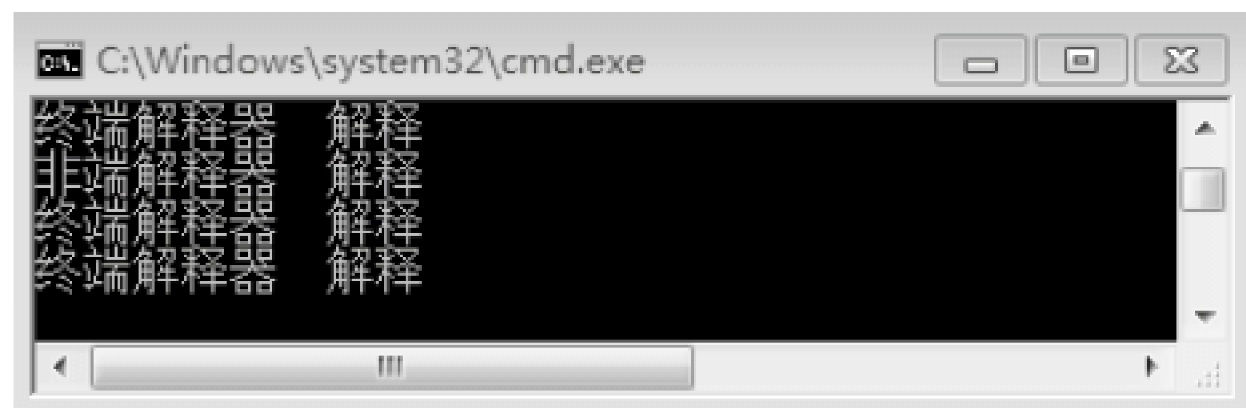


图 21.31　解释器模式

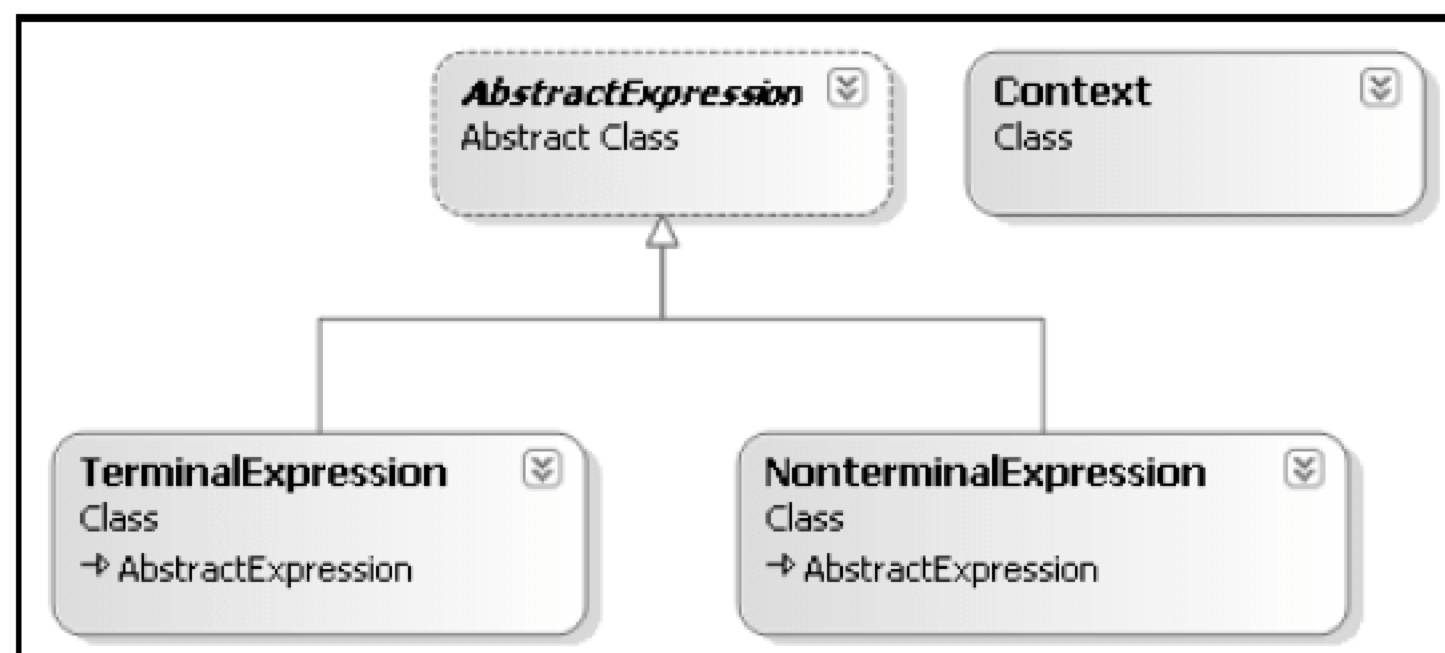
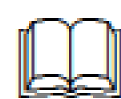


图 21.32　类关系设计图

说明：在图 21.32 中，AbstractExpression 是解释器的抽象类；TerminalExpression 是终端解释器类；NonterminalExpression 是非终端解释器类；Context 是解释内容类。

（2）解释器模式的适用性

当有一种语言需要解释执行，并且可将该语言中的句子表示为一个抽象语法树时，可使用解释器模式。而当存在以下情况时该模式效果最好：

- 文法简单。对于复杂的文法，文法的类层次将变得庞大而无法管理。此时类似语法分析程序生成器这样的工具是更好的选择。它们无须构建抽象语法树即可解释表达式，这样既可节省空间，又可节省时间。
- 效率不是一个关键问题。最高效的解释器通常不是通过直接解释语法分析树实现的，而是首先将它们转换成另一种形式。例如，正则表达式通常被转换成状态机。但即使在这种情况下，转换器仍可用解释器模式实现，该模式仍是有用的。

技巧：如果一种特定类型的问题发生的频率足够高，那么可能就需要将该问题的各个实例表述为一种简单语言中的句子。这样就可以构建一个解释器，该解释器通过解释这些句子来解决该问题。

（1）定义解释内容类 Context。代码如下：

```
class Context
{
    public static String ToString()
    {
        String context;
        return context= "解释";
    }
}
```

（2）定义终端解释器 TerminalExpression。代码如下：

```
class TerminalExpression    //终端解释器
{

    public static String get(String context){
            context = Context.ToString();
            String s = "终端解释器"+ context;
            return s;
    }
}
```

（3）定义非终端解释器 NonterminalExpression。代码如下：

```
class NonterminalExpression        //非终端解释器
{
    public static String get(String context){
            context = Context.ToString();
            String s = "非端解释器"+ context;
            return s;
    }
}
```

（4）在 main()方法中，首先将创建的各个终端、非终端解释器放入列表中，然后再循环列表，依次使用各个解释器解释指定的内容。代码如下：

```
import java.util.ArrayList;
import java.util.Iterator;
class Program
{
    public static void main(String[] args)
    {
        ArrayList list = new ArrayList();          //创建列表
        String context = null;
        list.add(TerminalExpression.get(context));
        list.add(NonterminalExpression.get(context));
        list.add(TerminalExpression.get(context));
        list.add(TerminalExpression.get(context));
```

```
            Iterator it = list.iterator();
            while(it.hasNext())
            {
                Object ob = it.next();
                System.out.println(ob);
            }
        }
}
```

秘笈心法

心法领悟 544：解释器模式的优点和不足。

- 易于改变和扩展文法。因为该模式使用类来表示文法规则，故可使用继承来改变或扩展该文法。已有的表达式可被增量式地改变，而新的表达式可定义为旧表达式的变体。
- 易于实现文法。定义抽象语法树中各个节点的类的实现大体类似，这些类易于直接编写（通常它们也可用一个编译器或语法分析程序生成器自动生成）。
- 复杂的文法难以维护。解释器模式为文法中的每一条规则至少定义了一个类，因此包含许多规则的文法可能难以管理和维护。可应用其他的设计模式来缓解这一问题；但当文法非常复杂时，其他的技术，如语法分析程序或编译器生成器更为合适。
- 增加了新的解释表达式的方式，使得实现新表达式“计算”变得容易。例如，可以在表达式类上定义一个新的操作以支持表达式的类型检查。如果需要经常创建新方式的解释表达式，则可以考虑使用 Visitor 模式以避免修改这些代表文法的类。

实例 545　迭代器模式　初级

光盘位置：光盘\MR\21\545

实用指数：★★★★

实例说明

什么是迭代器模式？笔者先通过一个实际生活中的例子进行介绍。

例如，乘坐长途客车回家，上车之后必须买票，如果忘记买票，司机或乘务员会提示乘客买票，最后的结果是客车上的每个乘客都买票了。乘车买票就是一个迭代器模式，司机或乘务员会把车上的所有乘客都遍历一遍，不放过任何一个不买票的乘客。

本实例中就应用迭代器模式实现遍历车上的每一名乘客，检查其是否买票。实例运行结果如图 21.33 所示。

关键技术

应用迭代器模式之前必须掌握迭代器模式的概念及适用性，下面分别介绍。

（1）迭代器模式的概念

迭代器模式提供了一种顺序访问一个聚合对象中各个元素的方法，而又不需暴露该对象的内部表示。本实例中类的关系如图 21.34 所示。

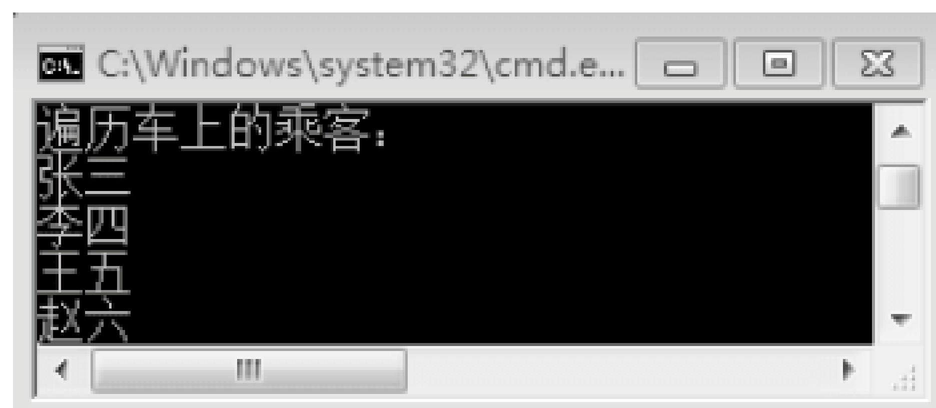


图 21.33　迭代器模式

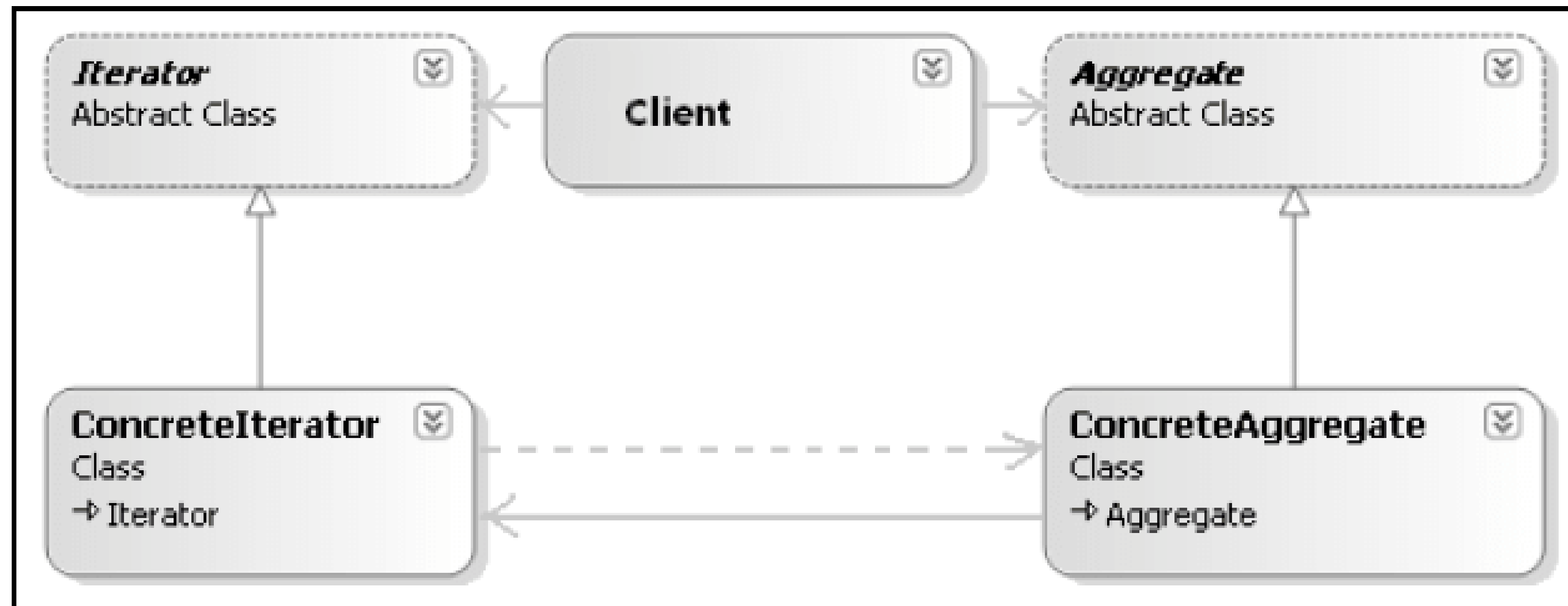


图 21.34　类关系设计图

说明：在图 21.34 中，Iterator 是迭代器抽象类；ConcreteIterator 是具体的迭代器类；Aggregate 是聚合对象抽象类；ConcreteAggregate 是具体的聚合对象类。

（2）迭代器模式的适用性

迭代器模式可用来访问一个聚合对象的内容而无须暴露其内部表示；支持对聚合对象的多种遍历；为遍历不同的聚合结构提供一个统一的接口（即支持多态迭代）。

（1）定义聚合对象抽象类 Aggregate。代码如下：

```
abstract class Aggregate                                    //聚合对象抽象类
{
    public abstract Iterator CreateIterator();              //创建迭代器
}
```

（2）定义具体的迭代器类 ConcreteIterator，继承 Iterator 类，并实现相关的方法。代码如下：

```
class ConcreteIterator extends Iterator                     //具体的迭代器类
{
    private ConcreteAggregate aggregate;                    //定义一个具体的聚合对象
    private int current = 0;
    Object ret = null;
    public ConcreteIterator(ConcreteAggregate aggregate)    //初始化时将具体的聚合对象传入
    {
        this.aggregate = aggregate;
    }
    public Object First()                                   //得到聚合的第一个对象
    {
      return aggregate.get(0);
    }
    public Object Next(int j)                               //得到聚合的下一个对象
    {
      ret = aggregate.get(j);
        return ret;
    }
    public boolean IsDone()                                 //判断当前是否遍历到结尾
    {
        return current >= aggregate.Count() ? true : false ;
    }
}
```

（3）定义迭代器抽象类 Iterator，并声明实现迭代功能的相关方法。代码如下：

```
abstract class Iterator                                     //迭代器抽象类
{
    public abstract Object First();
    public abstract Object Next(int j);
    public abstract boolean IsDone();
}
```

（4）定义具体的聚合对象类 ConcreteAggregate，继承 Aggregate 类。代码如下：

```
import java.util.ArrayList;
import java.util.Iterator;
class ConcreteAggregate extends Aggregate                   //具体的聚合对象
{
            ArrayList items = new ArrayList();              //创建列表，用于存放聚合对象
      public void Add(String str){
      items.add(str);
      }
      public Object get(int i){
            Object[] al = items.toArray();
            return al[i];
       }
       public Iterator CreateIterator()
      {
            return (Iterator) new ConcreteIterator(this);
       }
       public int Count()                            //返回聚合总个数
```

```
        {
                Object[] al = items.toArray();
                return al.length;
        }
}
```

（5）在 main()方法中，首先创建并初始化对象数组，然后创建迭代器对象，最后使用迭代器对象的相关方法遍历对象数组。代码如下：

```
import java.util.ArrayList;

class Program
{
    public static void main(String[] args)
    {
        ConcreteAggregate a = new ConcreteAggregate();    //对象数组
        ConcreteIterator i = new ConcreteIterator(a);     //声明了迭代器对象
        a.Add("张三");
        a.Add("李四");
        a.Add("王五");
        a.Add("赵六");
        System.out.println("遍历车上的乘客：");
        Object item = i.First();                          //迭代第一个对象
        System.out.println(item);
        for(int j = 1;j<a.Count();j++)
        {item = i.Next(j);
         System.out.println(item);
        }
    }
}
```

秘笈心法

心法领悟 545：迭代器模式的作用。

迭代器模式有 3 个重要的作用：

- 支持以不同的方式遍历一个聚合。例如，代码生成和语义检查要遍历语法分析树。代码生成可以按中序或前序来遍历语法分析树。迭代器模式使得改变遍历算法变得很容易，仅需用一个不同的迭代器的实例代替原先的实例即可。用户也可以自己定义迭代器的子类以支持新的遍历。
- 迭代器为遍历不同的结构提供统一的接口。
- 在同一个聚合上可以有多个遍历，每个迭代器保持自身的遍历状态。因此，用户可以同时进行多个遍历。

实例 546　观察者模式

光盘位置：光盘\MR\21\546

初级

实用指数：★★★★

什么是观察者（Observer）模式？笔者先通过一个实际生活中的例子进行介绍。

例如，有一位朋友推荐笔者和另外两个朋友购买一只股票，说短期内一定会涨，我们三位立即同意了：“好啊！有钱谁不赚呢？但有一个条件，你替我们看好股票，股票一有涨幅就通知我们。”这里用到的就是观察者模式，推荐股票的朋友可以看作是通知者，而笔者和另外两个朋友可以看作是观察者，股票一有涨幅，通知者就会通知观察者。

本实例中将应用观察者模式实现股票一有涨幅，通知者就通知 3 个股迷，运行结果如图 21.35 所示。



图 21.35　观察者模式

关键技术

应用观察者模式之前必须掌握观察者模式的概念及适用性，下面分别介绍。

（1）观察者模式的概念

定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。本实例中类的关系如图 21.36 所示。

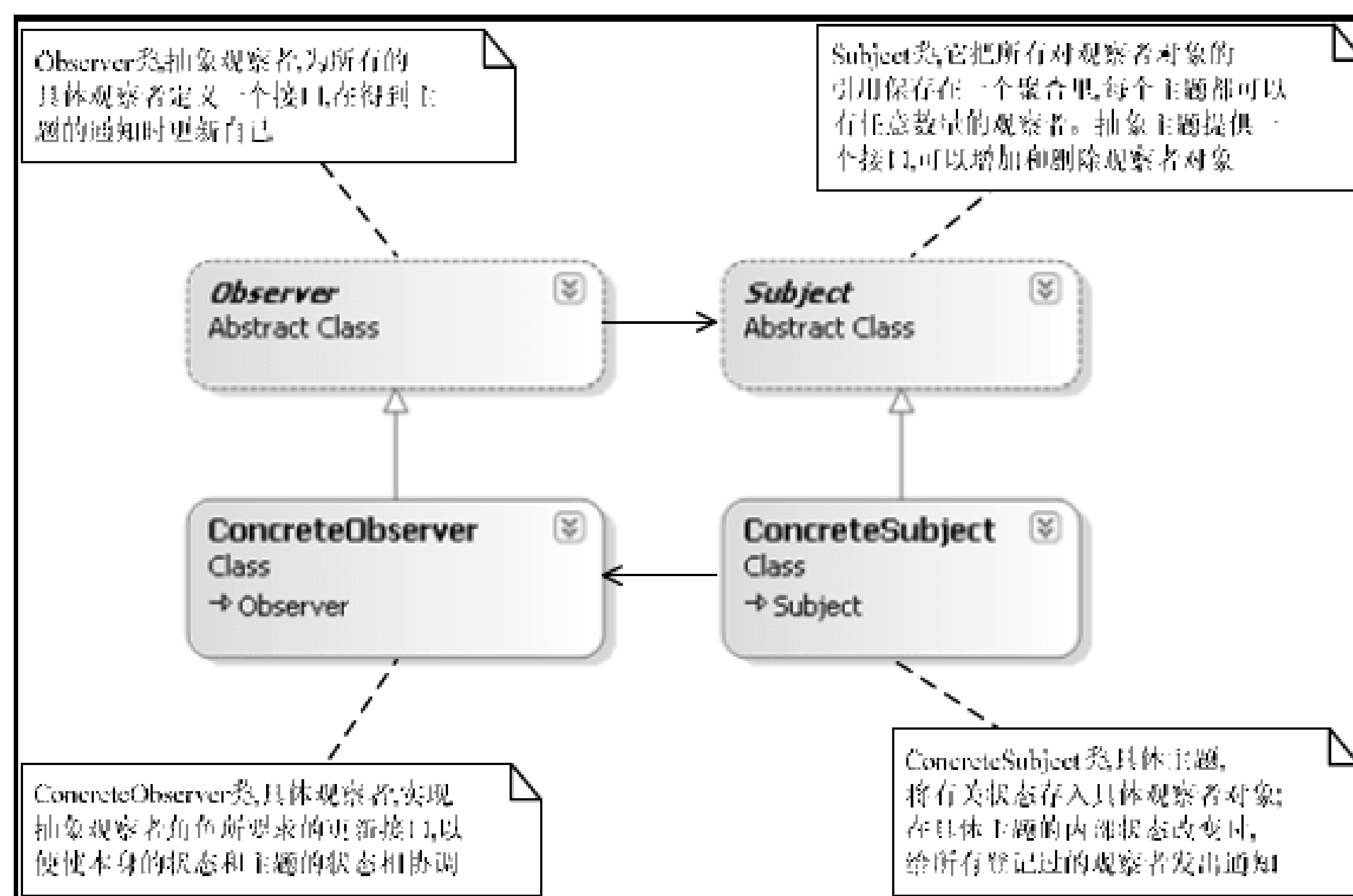


图 21.36　类关系设计图

（2）观察者模式的适用性

在以下任一情况下都可以使用观察者模式：

- 当一个抽象模型有两个方面，其中一个方面依赖于另一方面，将这二者封装在独立的对象中以使它们可以各自独立地改变和复用。
- 当对一个对象的改变需要同时改变其他对象，而不知道具体有多少对象有待改变。
- 当一个对象必须通知其他对象，而该对象又不能假定其他对象是什么。换言之，不希望这些对象是紧密耦合的。

（1）定义抽象主题类 Subject，主要实现把所有对观察者对象的引用保存在一个列表中，每个主题都可以有任意数量的观察者。抽象主题还提供一个接口，可以增加或删除观察者对象。代码如下：

```
abstract class Subject                                    //抽象主题（或通知者）
{
        private String str;
        ConcreteObserver cono = new ConcreteObserver(str);
        public void Attach(ConcreteObserver observer)  //增加观察者
        {
                cono.Add( observer);
         }
         public void Detach(Observer observer)           //删除观察者
         {
                cono.Remove(observer);
         }
         public void Notify(String str)                  //通知内容
         {
                        cono.Update(str);
         }
}
```

（2）定义具体主题类 ConcreteSubject，将有关状态存入具体观察者对象，在具体主题的内部状态改变时，给所有登记过的观察者发出通知。具体主题角色通常用一个具体子类实现。代码如下：

```
class ConcreteSubject extends Subject                      //具体主题（或通知者）
{
      private String subjectState;
```

```
            public String SubjectState(String subjectState)      //具体被观察者状态
            {
                    return subjectState;
            }
}
```

（3）定义抽象观察者类 Observer，为所有的具体观察者定义一个接口，在得到主题的通知时更新自身。代码如下：

```
abstract class Observer                                    //抽象观察者
{
    public abstract void Update(String str);               //更新方法
}
```

（4）定义具体观察者类 ConcreteObserver，实现抽象观察者角色所要求的更新接口，以便使自身的状态和主题的状态相协调。代码如下：

```
import java.util.ArrayList;
import java.util.Iterator;
class ConcreteObserver extends Observer                    //具体观察者
{
    private String name;
    private String observerState;
    private ConcreteSubject subject;
    ArrayList observers = new ArrayList();
    public ConcreteObserver( String name)
    {
        this.subject = subject;
        this.name = name;
    }
    public void Update(String str)
    {
        Iterator it = observers.iterator();
        while(it.hasNext())
        {
            observerState = it.next().toString();
            System.out.println("最新状态是："+" "+ observerState +"    "+ str);
        }
    }
    public ConcreteSubject Subject()
    {
         return subject;
    }
    public void Add(ConcreteObserver observer)
    {
      observers.add(observer.name);
    }
    public void Remove(Observer observer)
    {
            observers.remove(observer);
     }
}
```

（5）在 main()方法中，首先创建通知者，然后为通知者添加观察者，最后当改变状态时通知所有的观察者。代码如下：

```
class Program
{
    public static void main(String[] args)
    {
        ConcreteSubject s = new ConcreteSubject();         //创建通知者
        //添加观察者
        s.Attach(new ConcreteObserver( "股迷 A"));
        s.Attach(new ConcreteObserver( "股迷 B"));
        s.Attach(new ConcreteObserver( "股迷 C"));
        //改变状态
        String str = s.SubjectState("股票跌了");
        s.Notify(str);                                     //通知
    }
}
```

秘笈心法

心法领悟 546：观察者模式的优缺点。

- ❑ 支持广播通信。不像通常的请求，目标发送的通知不需指定其接收者，通知被自动广播给所有已向该目标对象登记的有关对象。目标对象并不关心到底有多少对象对自己感兴趣，它唯一的任务就是通知各个观察者。这给了用户在任何时刻增加和删除观察者的自由。处理还是忽略一个通知取决于观察者。
- ❑ 意外的更新。因为一个观察者并不知道其他观察者的存在，它可能对改变目标的最终代价一无所知。在目标上一个看似无害的操作可能会引起一系列对观察者以及依赖于这些观察者的对象的更新。此外，如果依赖准则的定义或维护不当，常常会引起错误的更新，这种错误通常很难捕捉。

实例 547	状态模式 光盘位置：光盘\MR\21\547	初级 实用指数：★★★★

实例说明

什么是状态（State）模式？笔者先通过一个实际生活中的例子进行介绍。

例如，笔者是个网迷，每天都上网到后半夜，后来家人强制规定笔者：10 点之前可以上网，10 点之后必须睡觉。这里可以应用状态模式，10 点之前是一个状态（上网），10 点之后是另一个状态（睡觉）。

本实例中应用状态模式实现了强制规定，运行结果如图 21.37 所示。



应用状态模式之前必须掌握状态模式的概念及适用性，下面分别介绍。

（1）状态模式的概念

当一个对象的内在状态改变时，允许改变其行为，这个对象看起来像是改变了它的类。本实例中类的关系如图 21.38 所示。

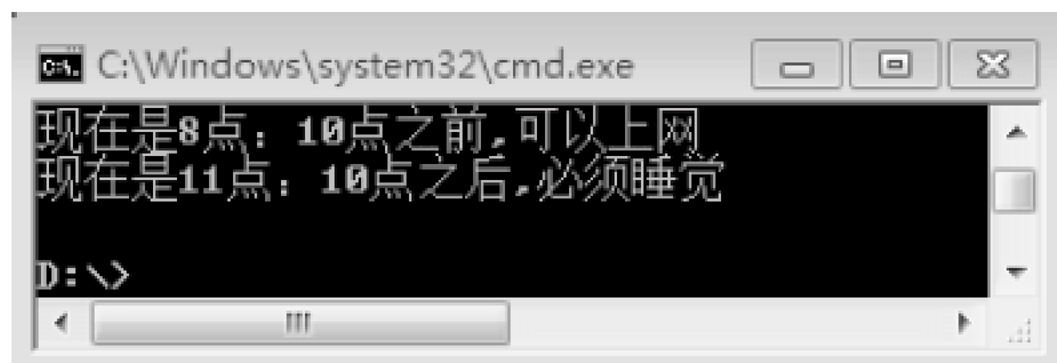


图 21.37　状态模式

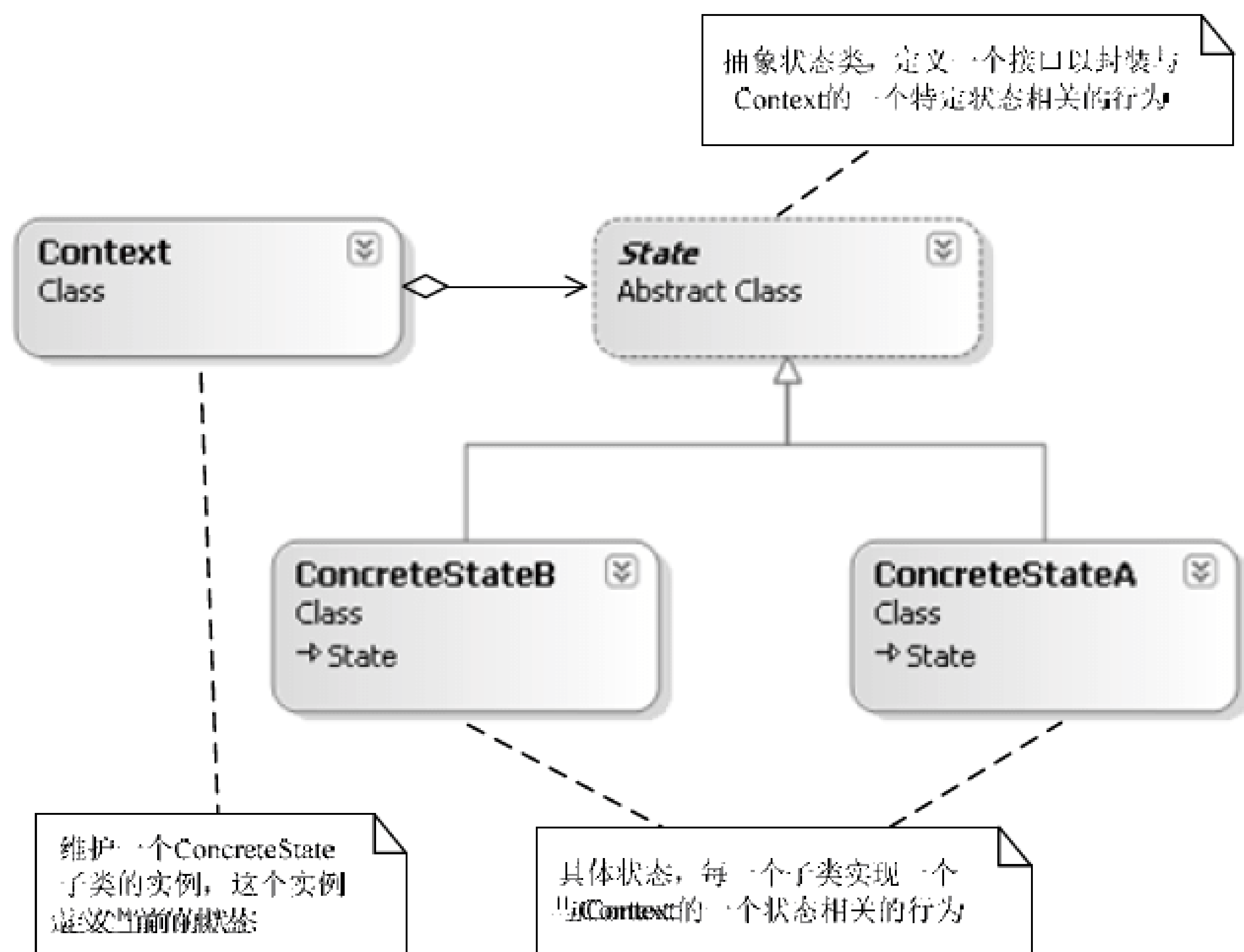


图 21.38　类关系设计图

（2）状态模式的适用性

在下面的两种情况下均可使用状态模式：

- ❑ 一个对象的行为取决于它的状态，并且必须在运行时根据状态改变其行为。

- 一个操作中含有庞大的多分支的条件语句，且这些分支依赖于该对象的状态。该状态通常用一个或多个枚举常量表示。通常，有多个操作包含这一相同的条件结构。状态模式将每一个条件分支放入一个独立的类中，使得用户可以根据对象自身的情况将对象的状态作为一个对象，这一对象可以不依赖于其他对象而独立变化。

（1）定义状态抽象类 State。代码如下：

```
abstract class State            //状态抽象类
{
    public abstract void Handle(Context context);
}
```

（2）定义具体状态类 ConcreteStateA，继承 State 类，实现一个与 Context 的一个状态相关的行为。代码如下：

```
class ConcreteStateA extends State                          //10 点之前的状态
{
    public void Handle(Context context)
    {
        if (context.Hour < 10)
        {
            System.out.println("现在是"+ context.Hour +"点：10 点之前，可以上网");
        }
        else
        {
            context.SetState(new ConcreteStateB());         //设置 ConcreteStateA 的下一个状态是 ConcreteStateB
            context.Request();
        }
    }
}
```

（3）定义具体状态类 ConcreteStateB，继承 State 类，实现一个与 Context 的一个状态相关的行为。代码如下：

```
class ConcreteStateB extends State                          //10 点之后的状态
{
    public void Handle(Context context)
    {
        if (context.Hour >= 10)
        {
            System.out.println("现在是" + context.Hour + "点：10 点之后，必须睡觉");
        }
        else
        {
            context.SetState(new ConcreteStateA());         //设置 ConcreteStateA 的下一个状态是 ConcreteStateB
            context.Request();
        }
    }
}
```

（4）定义 Context 类，维护一个具体状态类的实例，该实例定义当前的状态。代码如下：

```
class Context                           //状态上下文类
{
    private State state;                //当前状态
    public int Hour;                    //时间（小时）
    public Context()
    {
        state = new ConcreteStateA();   //设置 Context 的初始状态
    }
    public void SetState(State state)   //设置新状态
    {
        this.state = state;
    }
    public void Request()               //对请求进行处理，并设置下一状态
    {
        state.Handle(this);
    }
}
```

（5）在 main()方法中，分别设置当前时间是 8 点和 11 点，然后发出请求。代码如下：

```
class Program
{
    public static void main(String[] args)
    {
        Context c = new Context();      //设置 Context 的初始状态为 ConcreteStateA
        c.Hour = 8;                     //现在是 8 点
        c.Request();                    //请求
        c.Hour = 11;                    //现在是 11 点
        c.Request();                    //请求
    }
}
```

秘笈心法

心法领悟 547：状态模式的优点。

状态模式就是将特定状态的相关行为都加入一个对象中。由于所有与状态相关的代码都存在于 ConcreteState 中，所以通过定义新的子类可以很容易地增加新的状态和转换，即通过把各种状态转移逻辑分布到 State 的子类之间，来减少相互间的依赖。

实例 548 策略模式

光盘位置：光盘\MR\21\548

初级

实用指数：★★★★

实例说明

策略（Strategy）模式的用意是针对一组算法，将每个算法封装到具有共同接口的独立的类中，从而使得各算法可以相互替换。策略模式使得算法可以在不影响到客户端的情况下发生变化。

假设现在要设计一个类似 Pet Shop 4.0 电子商务网站的购物车（Shopping Cat）。一种最简单的情况就是把所有货品的单价乘上数量，但是实际情况复杂得多。例如，本网站可能对所有的教材类图书实行每本 1 元的折扣；对连环画类图书提供每本 7%的促销折扣；对非教材类的计算机图书有 3%的折扣；而对其余的图书则没有折扣。由于有这样复杂的折扣算法，使得价格计算问题需要系统地解决。

使用策略模式可以把行为和环境分割开来。环境类负责维持和查询行为，各种算法则在具体策略类（ConcreteStrategy）中提供。由于算法和环境独立开来，算法的增减、修改都不会影响环境和客户端。当出现新的促销折扣或现有的折扣政策出现变化时，只需要实现新的策略类，并在客户端登记即可。策略模式相当于“可插入式（Pluggable）的算法”。

本实例中应用策略模式实现了使用各种排序算法对人员列表进行排序，运行结果如图 21.39 所示。

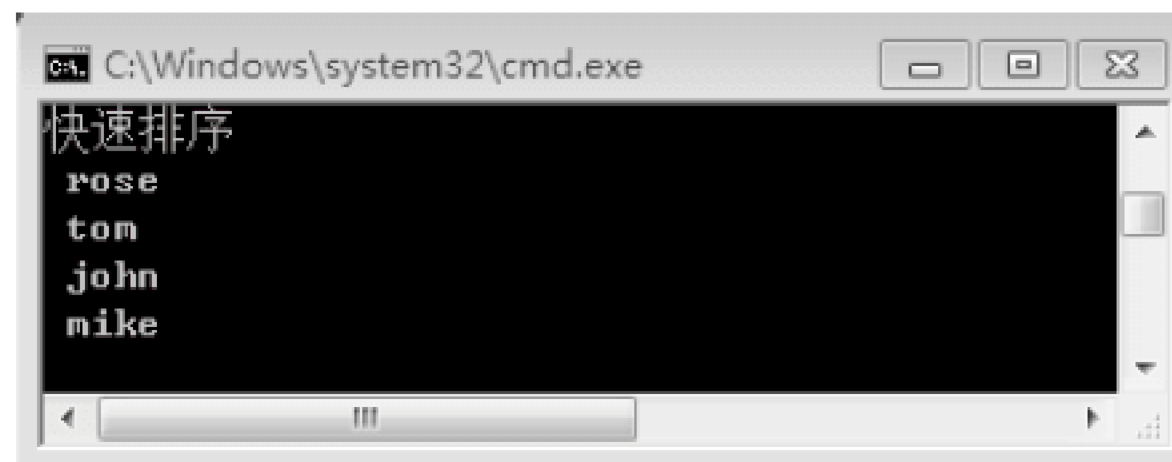


图 21.39　策略模式

下面详细介绍策略模式的优缺点。

（1）策略模式的优点

策略模式提供了管理相关算法族的方法。策略类的等级结构定义了一个算法或行为族，恰当使用继承可以把公共的代码移到父类中，从而避免重复的代码。

策略模式提供了替换继承关系的方法。继承可以处理多种算法或行为。如果不是用策略模式，那么使用算法或行为的环境类就可能会有一些子类，每个子类提供一个不同的算法或行为。但是，这样一来算法或行为的使用者就和算法或行为本身混在一起，决定使用哪一种算法或采取哪一种行为的逻辑就和算法或行为的逻辑混合在一起，从而不可能再独立演化。继承使得动态改变算法或行为变得不可能。

使用策略模式可以避免使用多重条件转移语句。多重转移语句不易维护，它把采取哪一种算法或采取哪一

种行为的逻辑与算法或行为的逻辑混合在一起，统统列在一个多重转移语句中，比使用继承的方法还要原始和落后。

（2）策略模式的缺点

客户端必须知道所有的策略类，并自行决定使用哪一个策略类。这就意味着客户端必须理解这些算法的区别，以便适时选择恰当的算法类。换言之，策略模式只适用于客户端知道所有的算法或行为的场合。

策略模式造成很多的策略类。有时可以把依赖于环境的状态保存到客户端中，而将策略类设计成可共享的，这样策略类实例可以被不同客户端使用。换言之，可以使用享元模式减少对象的数量。

（1）定义排序策略抽象类 SortStrategy，并声明排序方法 Sort()。代码如下：

```
import java.util.ArrayList;
abstract class SortStrategy                          //排序策略抽象类
{
    abstract public void Sort(ArrayList list);       //声明排序方法
}
```

（2）定义快速排序策略类 QuickSort，继承 SortStrategy 类，并实现 Sort()方法。代码如下：

```
import java.util.ArrayList;
class QuickSort extends SortStrategy                 //具体排序策略 1
{
    public void Sort(ArrayList list)
    {
        System.out.println("快速排序");
    }
}
```

（3）定义希尔排序策略类 ShellSort，继承 SortStrategy 类，并实现 Sort()方法。代码如下：

```
import java.util.ArrayList;
class ShellSort extends SortStrategy                 //具体排序策略 2
{
    public void Sort(ArrayList list)
    {

        System.out.println("希尔排序");
    }
}
```

（4）定义合并排序策略类 MergeSort，继承 SortStrategy 类，并实现 Sort()方法。代码如下：

```
import java.util.ArrayList;
class MergeSort extends SortStrategy                 //具体排序策略 3
{
    public void Sort(ArrayList list)
    {
       System.out.println("合并排序");
    }
}
```

（5）定义排序列表类 SortedList，该类主要实现添加元素、应用排序策略进行排序和显示排序结果。代码如下：

```
import java.util.ArrayList;
import java.util.Iterator;
class SortedList                                     //策略模式的关系
{
    private ArrayList list = new ArrayList();
    private SortStrategy sortstrategy;
    String name;
    public void SetSortStrategy(SortStrategy sortstrategy)
    {
        this.sortstrategy = sortstrategy;
    }
    public void Sort()                               //实现排序
    {
        sortstrategy.Sort(list);
    }
```

```
    public void Add(String name)                //添加元素
    {
        list.add(name);
    }
    public void Display()                       //显示排序结果
    {
        Iterator itr = list.iterator();
        while(itr.hasNext()){
        System.out.println(" " + itr.next() );
        }
    }
}
```

（6）在 main()方法中，首先创建自定义排序列表，然后为排序列表指定排序策略，最后排序并显示排序结果。代码如下：

```
public class StrategyApp
{
    public static void main(String[] args)
    {
        SortedList studentRecords = new SortedList();           //创建自定义排序列表
        studentRecords.Add("rose");                             //添加元素
        studentRecords.Add("tom");
        studentRecords.Add("john");
        studentRecords.Add("mike");
        studentRecords.SetSortStrategy(new QuickSort());        //为排序列表指定排序策略
        studentRecords.Sort();                                  //排序
        studentRecords.Display();                               //显示排序结果
     }
}
```

秘笈心法

心法领悟 548：策略模式与其他模式的关系。

策略模式与很多设计模式都有着广泛的联系。策略模式很容易和桥（Bridge）模式混淆，虽然二者结构相似，但却是为解决不同的问题而设计的。策略模式注重于算法的封装，而桥模式注重于分离抽象和实现，为一个抽象体系提供不同的实现。

实例 549 模板方法模式

初级

光盘位置：光盘\MR\21\549

实用指数：★★★★☆

实例说明

什么是模板方法（Template Method）模式？笔者先通过一个实际生活中的例子进行介绍。

例如，高考时的答卷（假设试卷上都是选择题），不管有多少考生，面对的选择题都是一样的，但是每个考生给出的选择答案未必一样。如果要将试卷及每个考生的答案封装成类，可以应用模板方法实现。将所有公共的内容都封装到基类中（例如选择题），只将不同的内容放到子类实现（例如每个考生的答案）。

本实例中应用模板方法模式实现学生 A 和学生 B 的答卷，运行结果如图 21.40 所示。

应用模板方法模式之前必须掌握模板方法模式的概念及适用性，下面分别介绍。

（1）模板方法模式的概念

模板方法模式就是定义一个操作中算法的骨架，而将一些步骤延迟到子类中，子类可以不改变该算法的结构即可重定义其某些特定步骤。本实例中类的关系如图 21.41 所示。

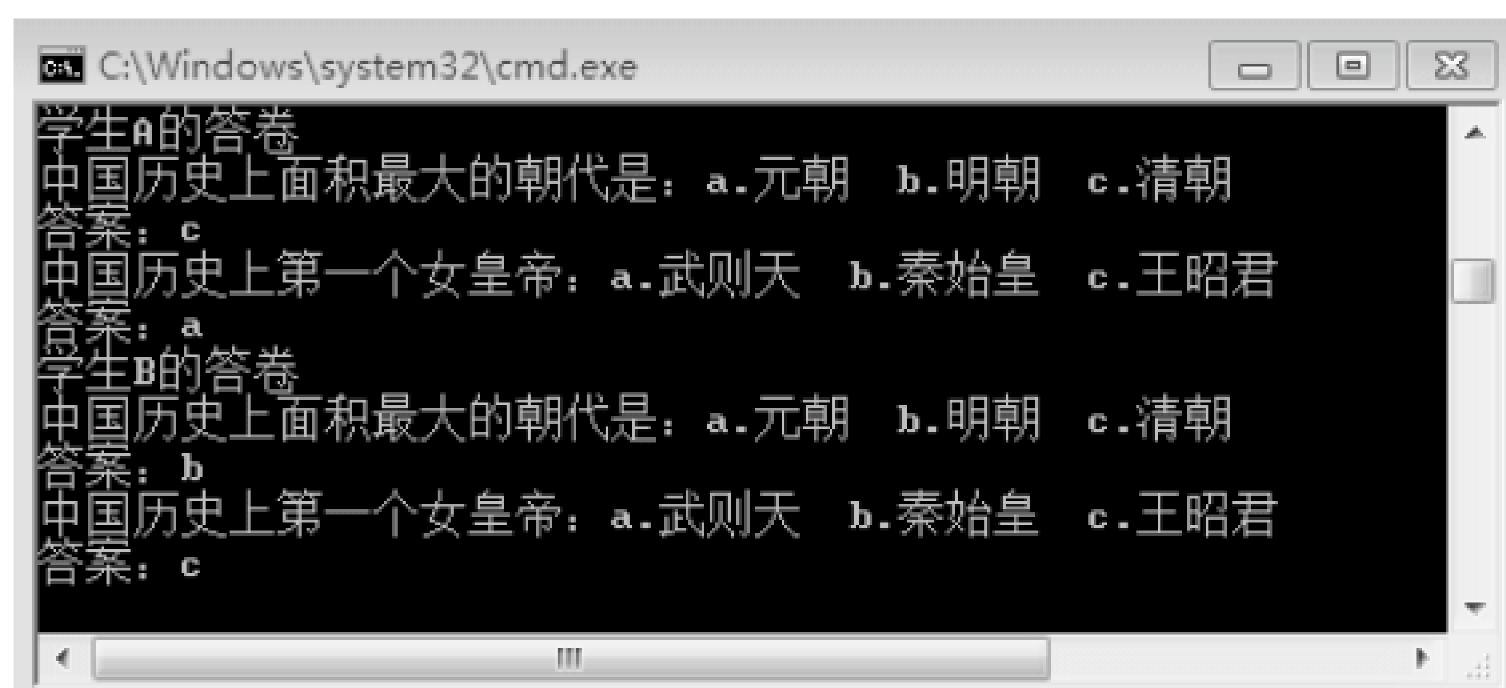


图 21.40　模板方法模式

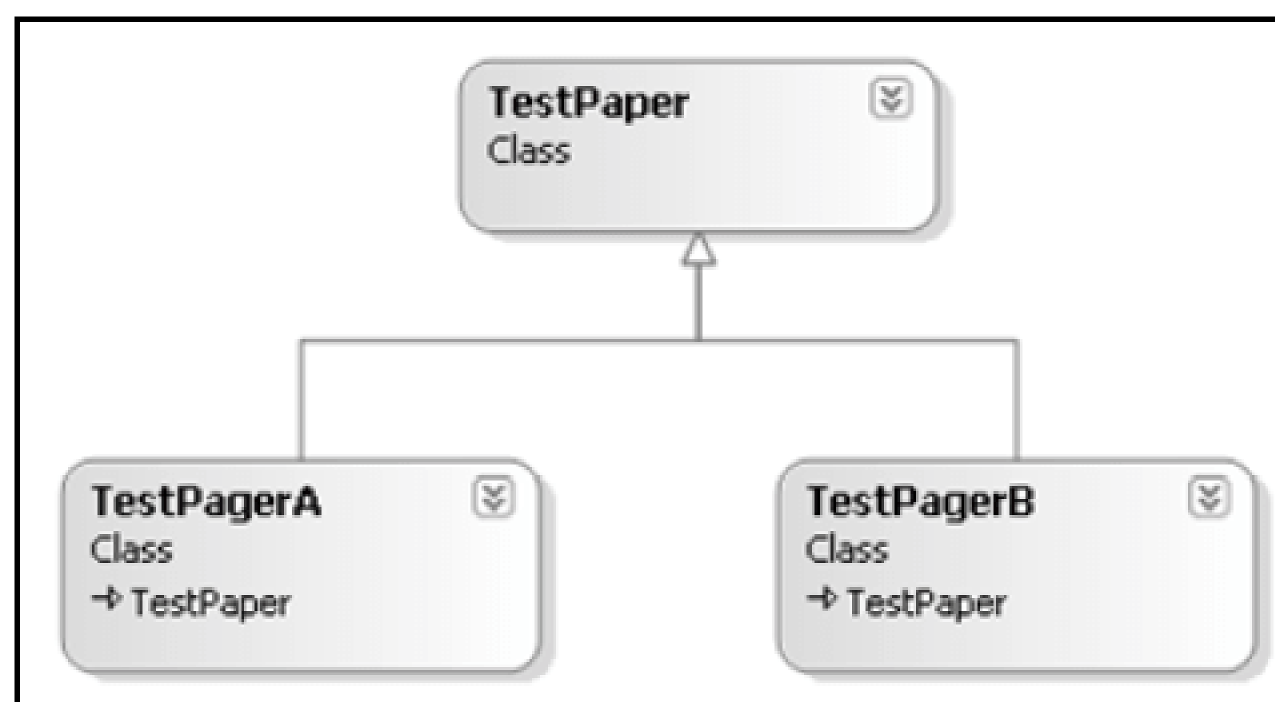


图 21.41　类关系设计图

说明：在图 21.41 中，TestPaper 是试卷基类；TestPagerA 是学生甲的试卷类；TestPagerB 是学生乙的试卷类。

（2）模板方法模式的适用性

模板方法模式适用于下列情况：

- 一次性实现一个算法的不变的部分，并将可变的行为留给子类实现。
- 各子类中公共的行为应被提取出来并集中到一个公共父类中以避免代码重复。具体做法是：首先识别现有代码中的不同之处，并且将不同之处分离为新的操作，最后用一个调用这些新的操作的模板方法替换这些不同的代码。
- 控制子类扩展。模板方法只在特定点调用 Hook 操作，这样就只允许在这些点进行扩展。

（1）定义试卷基类 TestPaper，该类中的 Answer1 和 Answer2 的具体功能在子类中实现。代码如下：

```
class TestPaper                          //试卷基类
{
    public void TestQuestion1()          //选择题 1
    {
        System.out.println("中国历史上面积最大的朝代是：a.元朝    b.明朝    c.清朝");
        System.out.println("答案：" + Answer1());
    }
    protected String Answer1()           //选择题 1 的答案
    {
        return "";
    }
    public void TestQuestion2()          //选择题 2
    {
        System.out.println("中国历史上第一个女皇帝：a.武则天    b.秦始皇    c.王昭君");
        System.out.println("答案：" + Answer2());
    }
    protected String Answer2()           //选择题 2 的答案
    {
        return "";
    }
}
```

（2）定义学生 A 的试卷类 TestPagerA，主要功能是重写两个选择题的答案。代码如下：

```
class TestPagerA extends TestPaper       //学生 A 的试卷
{
    protected String Answer1()           //重写选择题 1 的答案
    {
        return "c";
    }
    protected String Answer2()           //重写选择题 2 的答案
    {
        return "a";
    }
}
```

（3）定义学生 B 的试卷类 TestPagerB，主要功能是重写两个选择题的答案。代码如下：

```
class TestPagerB extends TestPaper       //学生 B 的试卷
{
```

```
    protected String Answer1()            //重写选择题 1 的答案
    {
        return "b";
    }
    protected String Answer2()            //重写选择题 2 的答案
    {
        return "c";
    }
}
```

（4）在 main()方法中，分别创建学生甲和学生乙的试卷类的实例，并实现两个学生的答卷。代码如下：

```
class Program
{
    public static void main(String[] args)
    {
        System.out.println("学生 A 的答卷");
        TestPaper studentA = new TestPagerA();
        studentA.TestQuestion1();
        studentA.TestQuestion2();
        System.out.println("学生 B 的答卷");
        TestPaper studentB = new TestPagerB();
        studentB.TestQuestion1();
        studentB.TestQuestion2();
    }
}
```

秘笈心法

心法领悟 549：什么是钩子操作。

钩子操作（Hook Operations）提供了默认行为，子类可以在必要时进行扩展。钩子操作在默认情况下通常是一个空操作。在此很重要的一点是模板方法应该指明哪些操作是钩子操作（可以被重定义），哪些是抽象操作（必须被重定义）。要有效地重用一个抽象类，子类编写者必须明确了解哪些操作是设计为有待重定义的。

实例 550	访问者模式 光盘位置：光盘\MR\21\550	初级 实用指数：★★★☆

实例说明

设置访问者（Visitor）模式的目的是将处理从数据结构中分离出来。利用该模式，可以在不改变某对象结构中各元素的类的前提下定义作用于这些元素的新操作。如果有比较稳定的数据结构，又有易于变化的算法，那么使用访问者模式就是比较适合的，因为该模式使得算法操作的增加变得更容易。

本实例将应用访问者模式实现不同的访问者分别访问列表中的元素，运行结果如图 21.42 所示。

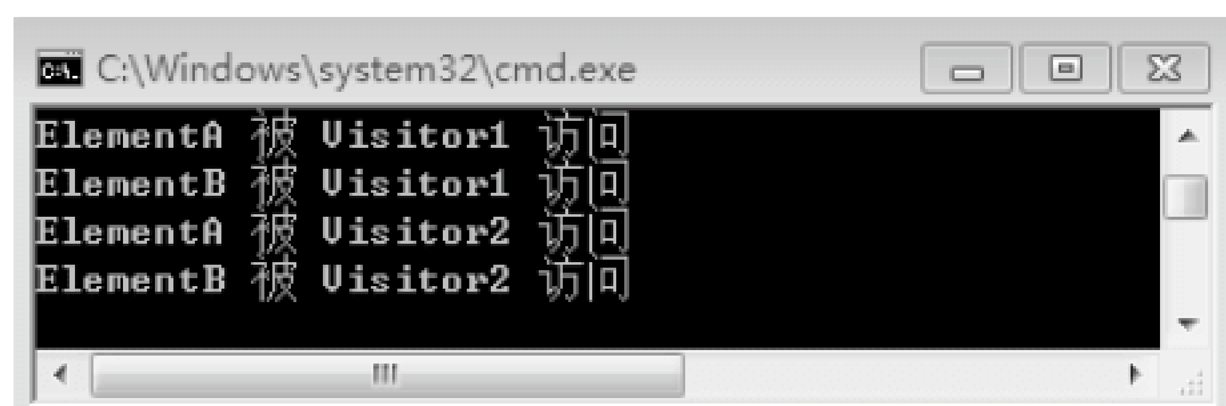


图 21.42　访问者模式

关键技术

在下列情况下使用访问者模式：

（1）一个对象结构中包含很多类对象，且有不同的接口，而想对这些对象实施一些依赖于其具体类的操作时。

（2）需要对一个对象结构中的对象进行很多不同的并且不相关的操作，又想避免让这些操作“污染”这些对象的类的。此时应用访问者模式可以将相关的操作集中起来定义在一个类中。当该对象结构被很多应用共享时，用访问者模式让每个应用仅包含需用到的操作。

（3）定义对象结构的类很少改变，但经常需要在此结构上定义新的操作。改变对象结构类需要重定义对所有访问者的接口，这可能需要很大的代价。如果对象结构类经常改变，那么还是在这些类中定义这些操作较好。

设计过程

（1）定义访问者抽象类 Visitor，并声明访问具体元素的各个方法。代码如下：

```
abstract class Visitor                                                              //访问者抽象类
{
    public abstract void VisitConcreteElementA(ConcreteElementA concreteElementA);  //访问具体元素 A
    public abstract void VisitConcreteElementB(ConcreteElementB concreteElementB);  //访问具体元素 B
}
```

（2）定义第一个具体访问者类 ConcreteVisitor1，并实现访问具体元素的各个方法。代码如下：

```
class ConcreteVisitor1 extends Visitor                                              //具体访问者类 1
{
        String name = "Visitor1";
    public void VisitConcreteElementA(ConcreteElementA concreteElementA)            //访问具体元素 A
    {
        System.out.println(concreteElementA.name+" 被 " + this.name+" 访问");
    }
    public void VisitConcreteElementB(ConcreteElementB concreteElementB)            //访问具体元素 B
    {
      System.out.println(concreteElementB.name+" 被 "+  this.name+" 访问");
    }
}
```

（3）定义第二个具体访问者类 ConcreteVisitor2，并实现访问具体元素的各个方法。代码如下：

```
class ConcreteVisitor2 extends Visitor                                              //具体访问者类 2
{     String name = "Visitor2";
      public void VisitConcreteElementA(ConcreteElementA concreteElementA)          //访问具体元素 A
      {
           System.out.println(concreteElementA.name+" 被 " + this.name+" 访问");
      }
            public   void VisitConcreteElementB(ConcreteElementB concreteElementB)  //访问具体元素 B
      {
           System.out.println(concreteElementB.name+" 被 "+  this.name+" 访问");
      }
}
```

（4）定义元素抽象类 Element，并声明 Accept()方法。代码如下：

```
abstract class Element                                                              //元素抽象类
{
    public abstract void Accept(Visitor visitor);
}
```

（5）定义具体元素类 ConcreteElementA，并实现 Accept()方法。代码如下：

```
class ConcreteElementA extends Element                  //具体的元素 A
{     String name = "ElementA";
    public void Accept(Visitor visitor)                 //充分利用双分派技术，实现处理与数据结构的分离
    {
        visitor.VisitConcreteElementA(this);
    }
    public void OperationA() {}                         //其他相关方法
}
```

（6）定义具体元素类 ConcreteElementB，并实现 Accept()方法。代码如下：

```
class ConcreteElementB extends Element                  //具体的元素 B
{     String name = "ElementB";
    public void Accept(Visitor visitor)
    {
        visitor.VisitConcreteElementB(this);
    }
    public void OperationB() {}
}
```

（7）定义 ObjectStructure 类，主要实现添加元素、删除元素及遍历元素接受访问。代码如下：

```
import java.util.ArrayList;
import java.util.Iterator;
class ObjectStructure
{
    private ArrayList elements = new ArrayList();       //创建元素列表
    public void Attach(Element element)                 //添加元素，元素数是固定的
    {
```

```
            elements.add(element);
        }
        public void Detach(Element element)                    //删除元素
        {
            elements.remove(element);
        }
        public void Accept(Visitor visitor)
        {
                Iterator it =elements.iterator() ;             //遍历元素列表
                while(it.hasNext()){
                ((Element)it.next()).Accept(visitor);
                }
        }
}
```

（8）在 main()方法中，首先创建对象结构列表，并添加元素，然后创建访问者对象，最后让元素接受访问。代码如下：

```
class Program
{
    public static void main(String[] args)
    {
        ObjectStructure o = new ObjectStructure();             //创建对象结构列表
        o.Attach(new ConcreteElementA());                      //添加元素
        o.Attach(new ConcreteElementB());
        ConcreteVisitor1 v1 = new ConcreteVisitor1();          //创建访问者对象
        ConcreteVisitor2 v2 = new ConcreteVisitor2();
        o.Accept(v1);                                          //接受访问
        o.Accept(v2);
    }
}
```

秘笈心法

心法领悟 550：双分派技术。

访问者模式中用到了双分派技术：首先在 main()方法中将具体的访问者作为参数传递给元素对象，完成第一次分派；然后在具体元素类的 Accept()方法中，将自己（this）作为参数传入访问者的相关方法，完成第二次分派。

21.5 网站开发架构模式

实例 551 MVC 框架在联系人管理网站中的应用

初级

光盘位置：光盘\MR\21\551

实用指数：★★★★☆

在 MVC（Model View Controller）框架下开发 Web 应用程序，与传统的 Web 技术有着很大的差异，它需要分别开发对应的模型、控制器和视图。本实例使用 MVC 框架开发一个联系人管理网站，实现了联系人信息的显示、添加。从本例中可以了解如何构建模型、如何实现控制器及如何自动创建对应的视图。联系人管理网站的主页面如图 21.43 所示。

在主页中单击“添加联系人”按钮，弹出添加联系人信息页面，如图 21.44 所示。

在添加联系人页面中输入新联系人信息后，单击“提交”按钮，即可将输入的内容保存到数据库中，并跳转到主页。



图 21.43　联系人管理网站的主页

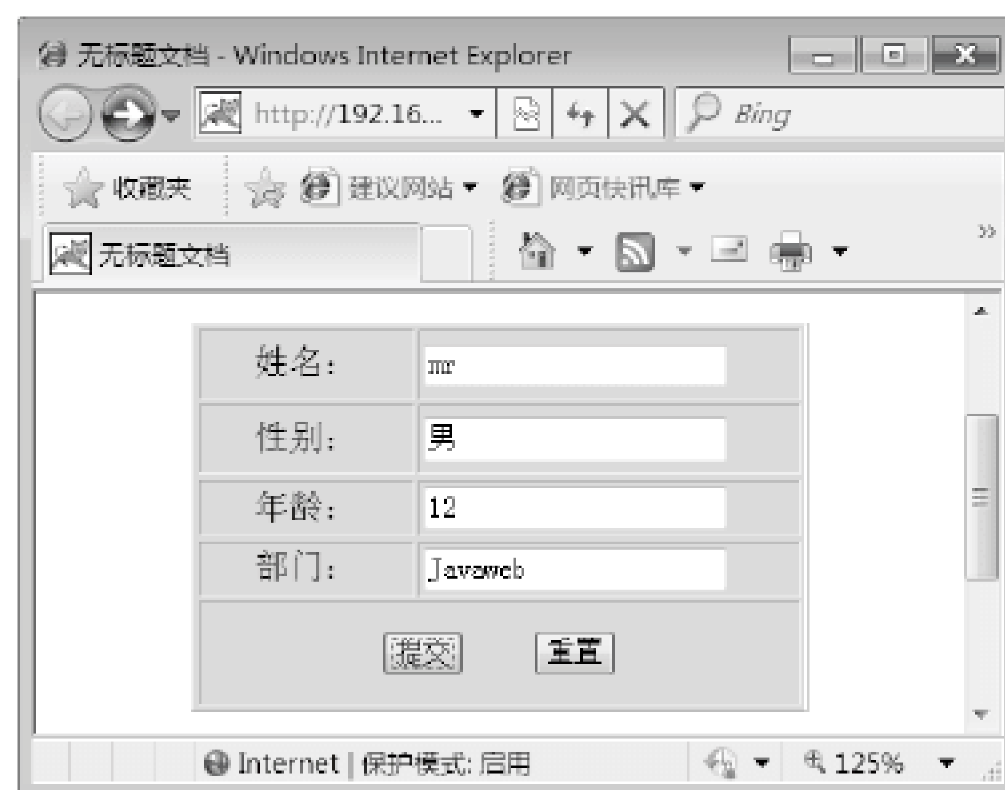


图 21.44　添加联系人信息页面

关键技术

1．什么是 MVC

MVC 设计模式将一般的应用程序根据功能的不同划分为模型、视图和控制器 3 部分。下面分别对这 3 部分做详细的介绍。

- 模型：在 MVC 设计模式中需要显示的数据。一般情况下相关代码从数据库中读取数据到模型，并保存模型的状态，提供数据访问方法及数据的维护。例如，对于 SQL Server 数据库 db_17 的表 tb_Linkman 来说，本实例 Models\Linkman.desinger.cs 文件中 tb_Linkman 类型的对象就是一个模型。该对象需要从数据表 tb_Linkman 中读取数据，并提供了对该数据表的查询、添加、修改、删除等方法。
- 视图：用来显示模型的用户界面，本实例的主页就是显示联系人信息列表的视图。
- 控制器：用来处理用户的输入或者交互命令，以便改变模型的状态，控制模型在视图上显示对应的数据。

2．MVC 之间的关系

在 MVC 框架中，模型、视图及控制器之间的相互关系如图 21.45 所示。

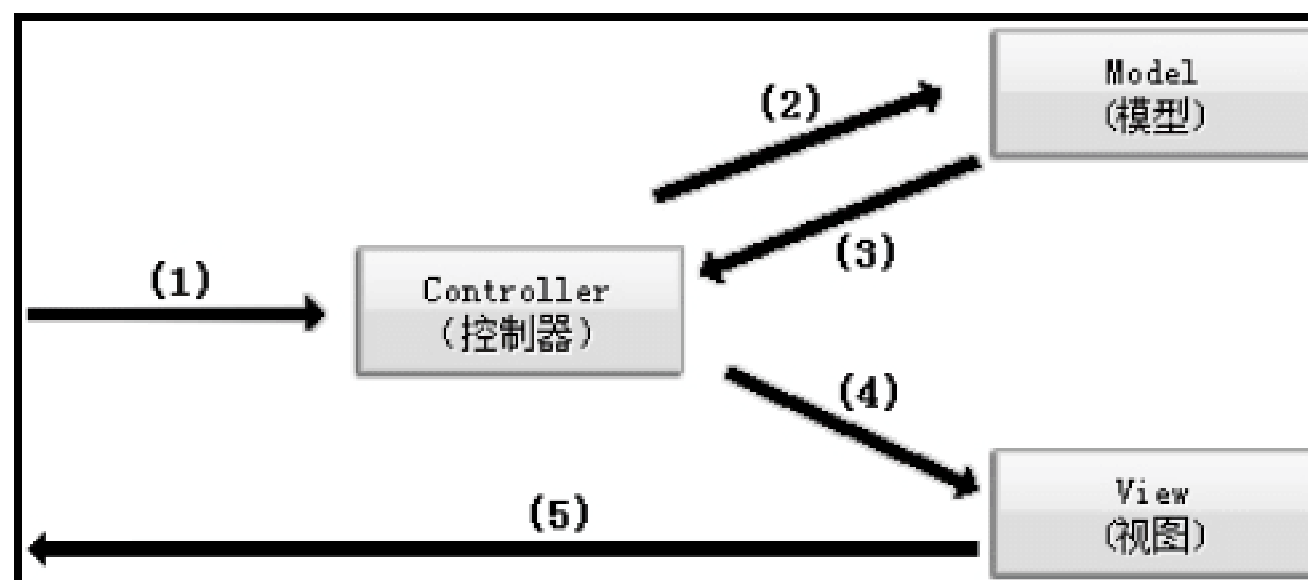


图 21.45　MVC 各组件之间的关系

从图 21.45 中可以看出，当用户在浏览器中输入浏览地址，到获得页面的反馈信息，通常需要经过以下 5 个步骤。

（1）当用户在浏览器中输入浏览地址、发送页面请求时，实际上是向控制器发送相关的命令。

（2）控制器接到用户的请求命令后，向模型请求获得相关的数据。

（3）模型将对应的数据返回给控制器。

（4）控制器再将模型返回的数据发送到指定的视图。

（5）指定的视图呈现数据。

从上述 5 个步骤中可以看出，控制器在其中扮演着十分重要的角色，它不仅处理用户的请求，还实现与模型之间的交互，对指定的视图发送相关的命令。

在开发基于 MVC 框架的联系人管理网站时，首先构建模型，其次实现控制器，最后根据控制器中定义的方法生成视图。

（1）创建模型层的代码 Person.java，封装相关属性，编写 get()和 set()方法。具体代码如下：

```
public class Person {
    private int id;
    private String name;
    private String sex;
    private String age;
    private String depart;
  //省略 get()和 set()方法
```

（2）创建控制器层的代码 personServlet.java，将处理数据的相关方法写入本文件中。具体代码如下：

```
public class personServlet extends HttpServlet {
	public personServlet() {
		super();
	}
	public void destroy() {
		super.destroy();
	}
	@SuppressWarnings("unchecked")
	public void doGet(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
	   Person pe=new Person();
	   PersonDAO pd=new PersonDAO();
		String name=new String(request.getParameter("name").getBytes("iso-8859-1"),"GBK");
		String sex=new String(request.getParameter("sex").getBytes("iso-8859-1"),"GBK");
		String age=new String(request.getParameter("age").getBytes("iso-8859-1"),"GBK");
		String depart=new String(request.getParameter("depart").getBytes("iso-8859-1"),"GBK");
		pe.setName(name);
	    pe.setAge(age);
	    pe.setSex(sex);
	    pe.setDepart(depart);
	    if(pd.Insert(pe))
	    {
	    	List<Person> list=pd.showall();
		request.setAttribute("list", list);
		request.getRequestDispatcher("showall.jsp").forward(request, response);
	    }
	}
	public void doPost(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
		response.setContentType("text/html");
		PrintWriter out = response.getWriter();
		out
				.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">");
		out.println("<HTML>");
		out.println("<HEAD><TITLE>A Servlet</TITLE></HEAD>");
		out.println("<BODY>");
		out.print("This is ");
		out.print(this.getClass());
		out.println(", using the POST method");
		out.println("</BODY>");
		out.println("</HTML>");
		out.flush();
		out.close();
	}
	public void init() throws ServletException {

	}
}
```

（3）创建视图层代码 index.jsp 和 showall.jsp，将数据的提交内容和录入模块写入 index.jsp 文件中，并且在 index.jsp 文件中写入 JS 代码，对数据进行非空验证；将显示模块代码写入 showall.jsp 中。index.jsp 的具体代码如下：

```
<script language="javascript">
    function yanzheng()
    {
      if(!document.form1.name.value)
        {
          alert("请输入姓名！ ");
          return false;
        }
      if(!document.form1.sex.value)
        {
          alert("请输入性别！ ");
          return false;
        }
      if(!document.form1.age.value)
        {
          alert("请输入年龄！ ");
```

```
                return false;
            }
        if(!document.form1.depart.value)
            {
                alert("请输入部门！ ");
                return false;
            }
    }

  </script>


<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>无标题文档</title>
<style type="text/css">
<!--
.STYLE1 {
        font-family: "宋体";
        font-weight: bold;
        font-size: 18px;
}
-->
</style>
</head>

<body>
<center> <br/><br/><br/>
  <form name="form1" action="ps" >
          <p align="center" class="STYLE1"> 注册信息页面：</p>
          <table width="282" border="1" bgcolor="#FFCCFF">
              <tr>
                  <td width="95" height="32"><div align="center">姓名：</div></td>
                  <td><input type="text" name="name" /></td>
              </tr>
              <tr>
                  <td width="95" height="32"><div align="center"> 性别：</div></td>
                  <td><input type="text" name="sex" /></td>
              </tr>
              <tr>
                  <td height="26"><div align="center">年龄：</div></td>
                  <td><input type="text" name="age" /></td>
              </tr>
              <tr>
                  <td height="25"><div align="center">部门：</div></td>
                  <td><input type="text" name="depart" /></td>
              </tr>
              <tr>
                  <td height="47" colspan="2"><div align="center">
                      <input name="submit" type="submit" value="提交" onClick="return yanzheng()"/>
                      <label>
                        
                      <input type="reset" name="Submit" value="重置">
                      </label>
                  </div></td>
              </tr>
          </table>
          <p align="center"><br>
      </p>
    <p align="center"> </p>
          <p align="center"> </p>
    <p align="center"><br>
    </p>
  </form>
  </center>
</body>
```

（4）showall.jsp 的具体代码如下：

```
<body>
<div align="center">
  <p align="right"><a href="index.jsp"></a></p>
```

```
  <p class="STYLE1">联系人列表</p>
  <table border="1" bgcolor="#FFCCFF">
    <tr >
      <td width="73"><div align="center"><strong>用户名 </strong></div></td>
     <td width="58"><div align="center"><strong>性别</strong></div></td>
     <td width="54"><div align="center"><strong>年龄 </strong></div></td>
     <td width="90"><div align="center"><strong>部门 </strong></div></td>
    </tr>
  </table>
  <table width="303" border="1">
    <%
      List<Person> list=(List)request.getAttribute("list");
      for(int i=0;i<list.size();i++){
   %>
    <tr >
      <td width="73"><div align="center"><%=list.get(i).getName()%></div></td>
     <td width="58"><div align="center"><%=list.get(i).getSex()%></div></td>
     <td width="54"><div align="center"><%=list.get(i).getAge()%></div></td>
     <td width="88"><div align="center"><%=list.get(i).getDepart()%></div></td>
    </tr>
    <%      }      %>
  </table>
  <p align="right"><a href="index.jsp" class="STYLE8">添加联系人</a></p>
</div>
</body>
```

秘笈心法

心法领悟 551：MVC 框架的优点。

易于单元测试：在 MVC 框架中，通过模型、视图、控制器，很好地分离了用户输入逻辑、业务逻辑和界面显示逻辑，因此非常容易实现 Web 应用程序的单元测试。

可扩展性强：MVC 框架中的各层代码组件可以被替换或者个性化。

实例 552 应用 MVC 架构开发简单计算器

光盘位置：光盘\MR\21\552

初级

实用指数：★★★★☆

Java Web 提供了很好的实现 MVC 模式的环境。通过在 JSP 页面中开发用户部件来实现视图；控制器的功能一般可以放在对应的逻辑功能代码（Servlet）中实现；模型通常对应系统的业务部分，一般包含业务逻辑、业务规则和数据访问层。MVC 可以和经典的 N 层结构配合使用。将用户显示（视图）从动作（控制器）中分离出来，提高了代码的重用性；将数据（模型）从对其操作的动作（控制器）分离出来，可以设计一个与后台存储数据无关的系统。就 MVC 结构的本质而言，它是一种解决耦合系统问题的方法。实现基于 MVC 的应用需要完成 4 个步骤，如图 21.46 所示。

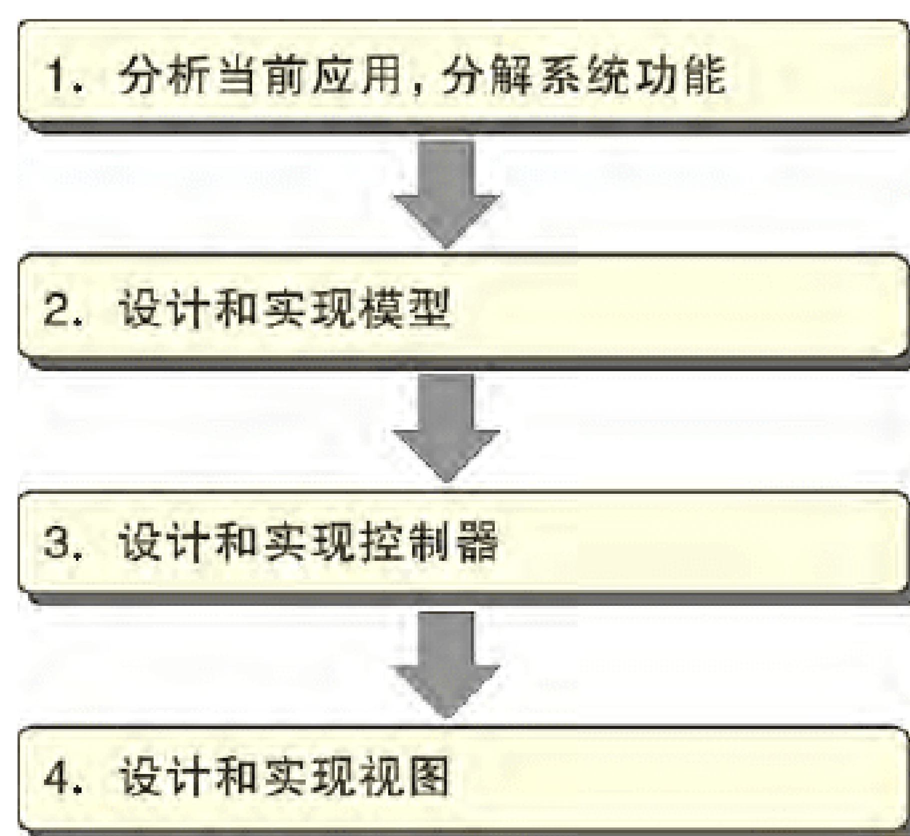


图 21.46　MVC 实现过程

（1）分析当前应用，分解系统功能。

分析当前应用问题，分离出系统的内核功能（Model）、系统的输入/输出（View）、系统的流程控制及行为控制等控制功能（Controller）三大部分。

（2）设计和实现模型。

设计模型部件使其封装应用功能、属性，提供访问显示数据的操作、控制内部行为的操作以及其他必要的操作接口。这部分的构成与具体的应用问题紧密相关。

（3）设计和实现控制器。

对于每个视图，实现将用户的请求映射到模型，并根据模型处理结果选择合适的视图显示。在模型状态的影响下，控制器使用特定的方法接受和解释这些事件。控制器的初始化建立起与模型、视图的联系（这里一般会用观察者模式）并且启动事件处理机制，事件处理机制的具体实现方法依赖于界面的工作平台。

（4）设计和实现视图。

设计每个视图的显示形式，从模型中获取数据，并将其显示在屏幕上。模型提供发送用户请求给控制器，并且提供允许控制器选择的视图。

MVC 并没有明确的定义，仅代表一种软件设计思想，所以在不同的应用环境下，可能有不同的实现方式。只有深刻理解其思想，结合实际情况，才能构建合理的应用。

关键技术

下面详细讲解 MVC 在 Java Web 下的设计原理。

MVC 架构通过区分各个层，允许组成每个层的各个组件间松散地耦合。这使得程序开发更加灵活，并且可以减少重复性代码，实现代码重用。MVC 的设计架构如图 21.47 所示。

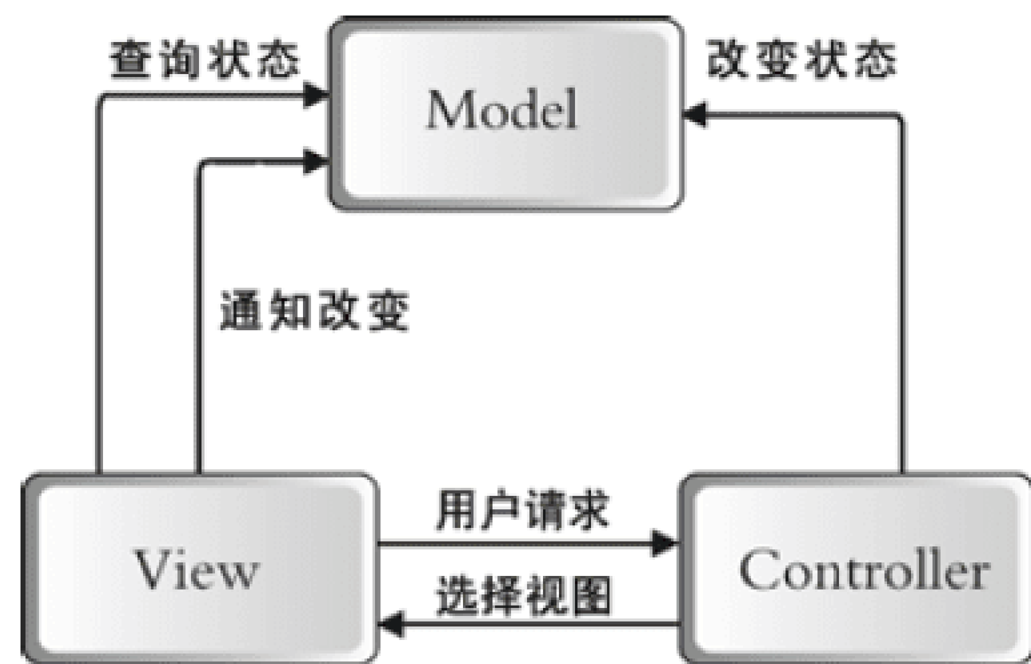


图 21.47　MVC 设计架构图

模型组件表示应用程序的数据，并包括这些数据的访问和修改的业务规则，独立于用户界面和 I/O 操作。

视图组件是用户看到并与之交互的界面，主要负责从模型访问数据，指定如何表示数据，并且当模型改变时，维护表示的一致性。此外，视图也负责把用户动作传递给控制器。

控制器组件定义应用程序的行为，解释用户动作，并把它映射为模型执行的过程。它负责模型和视图之间的交互，控制对用户输入的响应方式和流程，主要包括两个动作：一方面将用户的请求分发到相应的模型，另一方面将模型的改变及时反映在视图上。

- 视图（View）的实现

在 Java Web 开发环境中设计视图十分便捷。视图由 JSP 文件实现。它可以是最简单 HTML 的控件，可以通过一些提供的标签对页面的内容进行设定以及使用一些 CSS 样式对页面的布局进行设计。视图与各模块中的 Bean 文件相对应。

- 控制器（Controller）的实现

Controller 控制器是 Model 与 View 之间沟通的桥梁，它可以分派用户的请求并选择恰当的视图予以显示，也可以解释用户的输入并将它们映射为模型层可执行的操作。每个 JSP 页面都有一种机制，将页面中控件所要调用的方法在一个与其分离的类中实现。

- 模型（Model）的实现

Model 对象代表了商业规则和商业数据，单个模型代表问题域中的某个对象，或叫做实体。所以模型要封装系统的应用功能和应用属性以及提供访问显示数据的操作、控制内部行为的操作以及其他必要的操作接口。模型的构成与具体的应用问题紧密相关，通常包括数据访问、商务逻辑和商务规则。在 JSP 中，简单的模型可以方便地用自动代码生成工具实现。

（1）建立模型模块，创建 JavaBean 代码，主要实现对一些属性的封装。实现代码如下：

```
public class ComputerBean {
        double numA;
        double numB;
        String operator="+";
        double result;
//省略 get()和 set()方法
}
```

（2）模块（视图）设计完成后，开始设计控制器（Controller）。创建一个类文件，在其中编写相关的计算

代码。实现代码如下：

```
public class RComputer extends HttpServlet{
    public void init(ServletConfig config)throws ServletException{
        super.init(config);
    }
    public void doPost(HttpServletRequest request,HttpServletResponse response)
                                throws ServletException,IOException{
        ComputerBean data=null;
        HttpSession session=request.getSession(true);
        try{
            data=(ComputerBean)session.getAttribute("number");
            if(data==null){
                data=new ComputerBean();
                session.setAttribute("number", data);
            }
        }
        catch(Exception e){
            data=new ComputerBean();
            session.setAttribute("number", data);
        }
        double one=Double.parseDouble(request.getParameter("numA"));
        double twe=Double.parseDouble(request.getParameter("numB"));
        String oper=request.getParameter("operator");
        double result=0;
        if(oper.equals("+")){
            result=one+twe;
        }
        else if(oper.equals("-")){
            result=one-twe;
        }
        else if (oper.equals("*")){
            result=one*twe;
        }
        else if(oper.equals("/")){
            result=one/twe;
        }
        data.setNumA(one);
        data.setNumB(twe);
        data.setOperator(oper);
        data.setResult(result);
        RequestDispatcher dis=request.getRequestDispatcher("result.jsp");
        dis.forward(request, response);
    }
    public void doGet(HttpServletRequest request,HttpServletResponse response)
            throws ServletException,IOException{
        doPost(request,response);
    }
}
```

（3）完成模块（Model）和控制器（Controller）的创建后，开始设计 View 视图。本例共包含两个 Web 窗体，分别为计算页面和计算结果页面，如图 21.48 和图 21.49 所示。

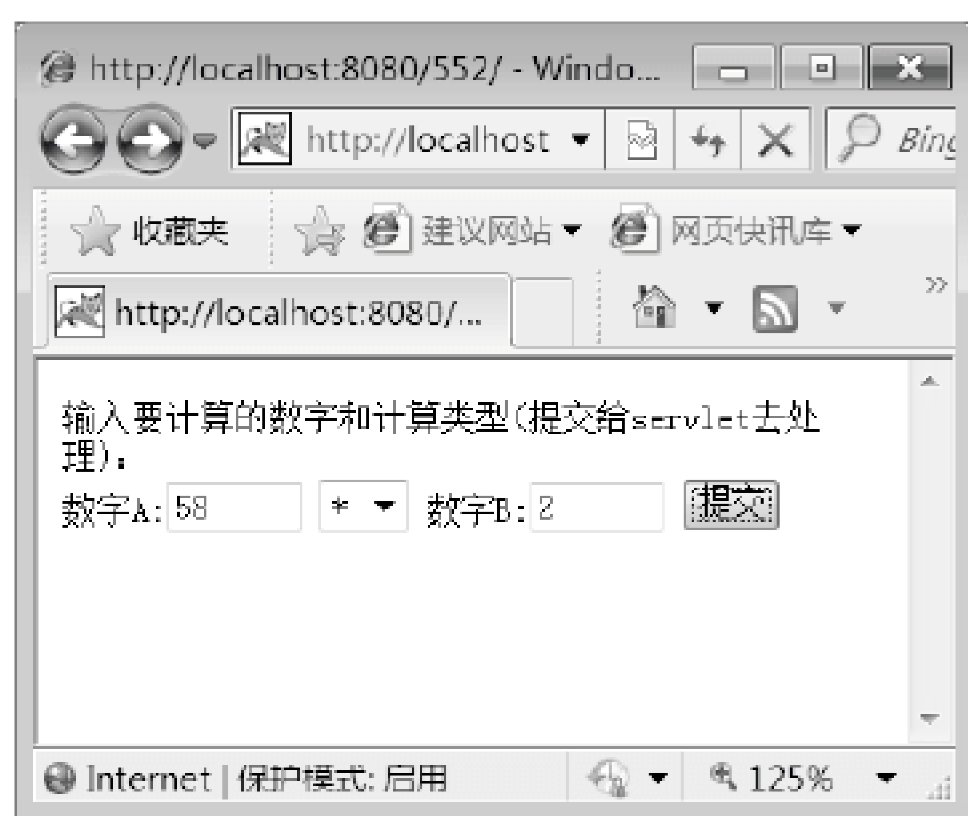


图 21.48　Default.aspx 页面设计结果

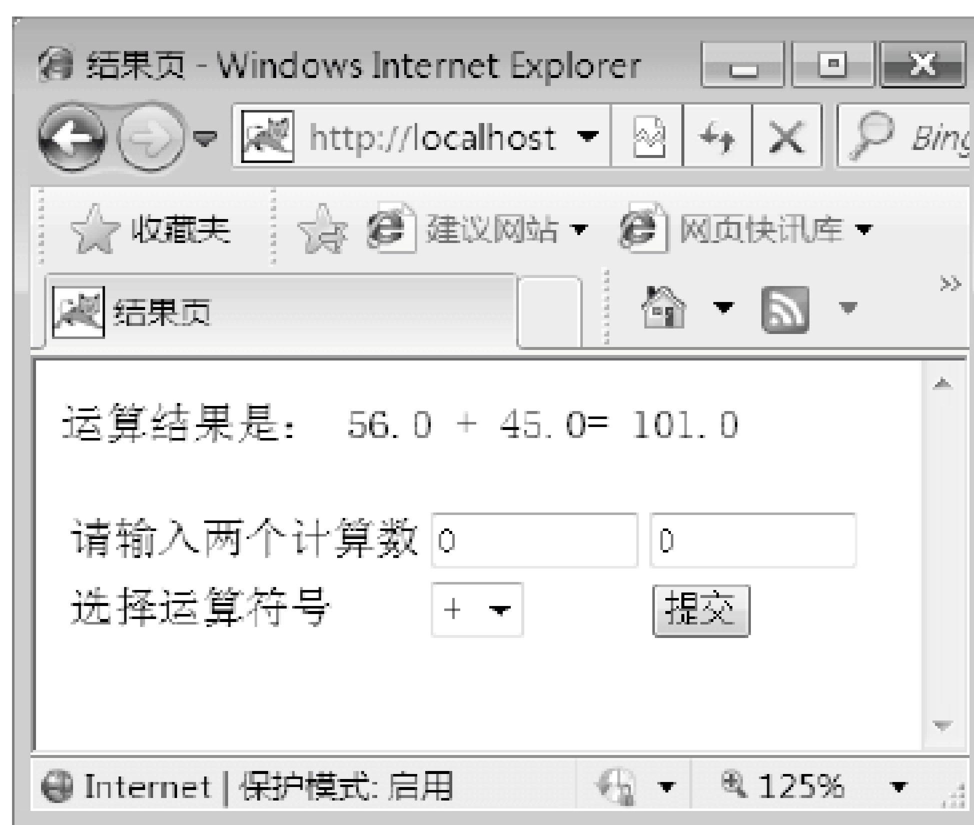


图 21.49　结果页面

（4）创建 input.jsp 和 result.jsp 页面，实现输入数字的文本框和选择运算符的下拉列表，以及“提交”按钮。Input.jsp 页面代码如下：

```
<FORM action="rcomputer" Method="post" >
  <P>输入要计算的数字和计算类型（提交给 Servlet 去处理）：
  <BR>数字 A：<Input type=text name="numA" value=0 size=6>
  <select name="operator">
  <option value="+">+
  <option value="-">-
  <option value="*">*
  <option value="/">/
  </select>
         数字 B：<Input type=text name="numB" value=0 size=6>
  <Input type=submit value="提交">
</FORM>
```

（5）result.jsp 页面代码如下：

```
<body>
  运算结果是：
    <jsp:useBean id="number" type="fe.zx.ComputerBean"scope="session"/>
    <jsp:getProperty name="number" property="numA" />
      <jsp:getProperty name="number" property="operator" />
    <jsp:getProperty name="number" property="numB" />=
   <jsp:getProperty name="number" property="result" />
   <form action="rcomputer" method="post">
   <table><tr><td>请输入两个计算数</td>
   <td><input type="text" name="numA" value="0"size=10></td>
   <td><input type="text" name="numB" value="0"size=10></td> </tr>
   <tr><td>选择运算符号</td>
   <td><select name="operator">
  <option value="+">+<option value="-">-
  <option value="*">*<option value="/">/
  </select> </td>
  <td> <Input type=submit value="提交"></td> </tr></table> </form>
  </body>
```

（6）配置 web.xml 文件，对 Servlet 的内容进行配置。实现代码如下：

```
<welcome-file-list>
    <welcome-file>input.jsp</welcome-file>
  </welcome-file-list>
<servlet>
    <servlet-name>handle</servlet-name>
    <servlet-class>fe.zx.RComputer</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>handle</servlet-name>
  <url-pattern>/rcomputer</url-pattern>
</servlet-mapping>
</web-app>
```

心法领悟 552：在 web.xml 文件中配置首页。

如要配置程序的默认首页，在 web.xml 文件中修改<welcome-file-list>标签中的页面数据即可。具体代码如下：

```
<welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

第7篇

综合应用篇

- 第22章　网站设计与网页配色
- 第23章　Java Web典型项目开发案例

第22章

网站设计与网页配色

- ⏭ 企业网站
- ⏭ 电子商务类
- ⏭ 搜索引擎类
- ⏭ 生活资讯类
- ⏭ 娱乐类网站
- ⏭ 供求信息类
- ⏭ 其他应用

22.1　企 业 网 站

在互联网上有无数个不同类型的网站，企业网站便是其中颇具代表性的一种。与传统媒体不同，企业网站有着鲜明的服务特色及功能。企业可以通过自己的网站向世界展示自己，发布本企业的信息，宣传企业文化、经营理念，加强与客户的联系等。那么如何将企业网站设计得更合理、美观、实用呢？下面通过几个实例详细介绍企业网站设计与网页配色中的常用方法和技巧。

实例 553　汽车销售网

光盘位置：光盘\MR\22\553

初级

实用指数：★

实例说明

汽车销售网是一种服务性网站。为了更好地引导消费者查询汽车行情、了解汽车产品，其风格应以其品牌形象为主要诉求点，以突出其企业文化、特质以及产品信息。一汽大众作为东北首屈一指的重工业龙头企业，其网站在结构上阅读性强；在色彩上注目性高；在制作技术上，合理地运用 Flash 技术将整个企业的企业文化与经营理念烘托得淋漓尽致，如图 22.1 所示。



图 22.1　一汽大众网站

关键技术

一个网站的好坏取决于很多因素。文字与图片是网站构成的基础，其摆放位置、大小、色彩等都影响着整个网站设计的成败。文字与图片的关系可以考虑两种情况：一种是文字印在图片上，此时除了保证文字在图片上容易辨认之外，还应认真推敲图片的内容、视觉特征、构图及色调，从而选择合适的位置安排文字，使文字画面形成统一的整体而不破坏图片的视觉效果；另一方面，在图片较多的情况下，则要进行合理的图文排版。将数张图片整齐有序地进行排列，可以产生鲜明的理性感；如果将图片分散进行组合，则会给用户自由、明快、不拘谨、版面轻松愉快的感觉。一汽大众网站采用了理性化的图片处理方式，突出了其企业的理性风格。

设计过程

（1）确定网站的整体风格

随着生活水平的提高，拥有汽车不再可望而不可即，汽车销售业也因此竞争更加激烈。为了更好地展示企业的品牌形象，推销自己的汽车产品，大多数企业都以事实、价值为基础，以形象塑造为主，制作大气、稳定、可靠、诚信、页面整洁、图文清晰、企业文化韵味浓厚的网站，从而起到建立客户忠诚度、增加客户价值的作用，并且拓展、建立、保持并强化了客户关系。

（2）确定页面的框架结构

企业网站在宣传其形象的同时，也在介绍其产品。为了使广大浏览者阅读更舒适，在页面的框架设计上通常采用引导性强、结构清晰的构架形式。其中，分栏式结构被大多数设计师运用。

注意： 不是所有的企业网站都可以运用分栏式结构设计，它并不是“教条式”的，应该根据网站的信息内

容、相关网站资料等决定采用什么样的结构进行设计。

（3）网页色彩的运用

企业网站通常会选择明亮的色彩，如蓝色、白色等，使整个网站看起来大气恢弘，从而提高企业的诚信度。企业网站的色彩应主要从体现企业文化的角度进行设计。另外，企业标准色也是确立其网页主体色的要素之一。

秘笈心法

心法领悟 553：汽车销售网站特色之处。

汽车销售网站注重于企业形象、品牌形象、主打产品形象的确立，可以通过风格简约、导航清晰、Flash 制作明快、品牌标示鲜明等特点吸引消费者的眼球。

实例 554	医药连锁网 光盘位置：光盘\MR\22\554	初级 实用指数：★★

实例说明

医药类网站是企业网站中的一种。这类网站通常运用简洁的结构、高清晰度的插图以及纯净、光亮的蓝色或者绿色等烘托专业、诚信、洁净、环保的整体网站风格。如美信 USA 国际连锁网就是运用蓝色为网站的主体色，体现出了医药的行业特点，如图 22.2 所示。



图 22.2　美信 USA 国际连锁网

关键技术

在色彩的运用过程中，使用一种色彩是比较保险的一种配色方式。这里是指先选定一种色彩，然后调整其透明度或者饱和度（通俗地讲，就是将色彩变淡或者加深）并应产生新的色彩，用于网页。这样的页面看起来色彩统一，富有层次感。使用统一的颜色，可以形成网站的整体风格。

注意： *在运用一种颜色进行配色时，对于色彩的落差、层次的把握是这种色彩表达方式成功与否的决定因素。*

设计过程

（1）确定网站的整体风格

由于医药类网站是一种专业性很强的网站，这就要求该类网站必须符合企业本身的行业特点。首先应该诚信，让浏览此类网站的消费群体认可。整个页面整体感觉要洁净、环保、专业性强，才能使整个网站有医药的气息，从而取得消费者的认可。如美信 USA 国际连锁网站既体现出其鲜明的企业形象，又对整体网站氛围得以全面诠释。

（2）确定页面的框架结构

规律框架结构与无规律框架结构相结合，越来越多地被设计师运用到网站设计当中，不但能够适应信息相对较多的网站，也可以使网站的整个页面看起来更加灵活。简洁清晰的页面构架方式更加实用，再配上可以表达主题内容的插图和合理的色彩搭配，将网站的整体氛围表现了出来。如美信 USA 国际连锁网站的框架结构简单、明快、引导性强，不失为一个较好的网站。

（3）网页色彩的运用

通过对行业特性的了解，在设计医药类网站时，通常运用的颜色是纯净、明亮的，如蓝色与绿色。蓝色代表诚信、稳重，可以使浏览者产生可信赖的感觉，冷色的蓝还可以表现一种清爽、明快的感觉。绿色则代表生命。再结合本例“关键技术”中讲到的“一种色彩”的表现形式，即可形成统一的、有层次感、洁净、环保、专业、诚信的网站整体风格。

秘笈心法

心法领悟 554：在医药网站配色中多运用绿色。

绿色与人类息息相关，是永恒的欣欣向荣的自然之色，代表了生命与希望、和平与安全、发展与生机、舒适与安宁、松弛与休息，所以建议在医药网站配色中多运用绿色。

实例 555	硬件产品网 光盘位置：光盘\MR\22\555	初级 实用指数：★★

实例说明

硬件产品网站在设计时都是以延续其企业理念与品牌形象为依据来寻找设计的切入点。在设计风格上注重稳重、诚信、大气、理性、简洁、明快。如联想网站在设计上遵循其行业特点，运用大量精致的广告图片来宣传自己的产品，充分突出了其整体风格，不失为最具代表性的大型硬件产品网站之一，如图 22.3 所示。

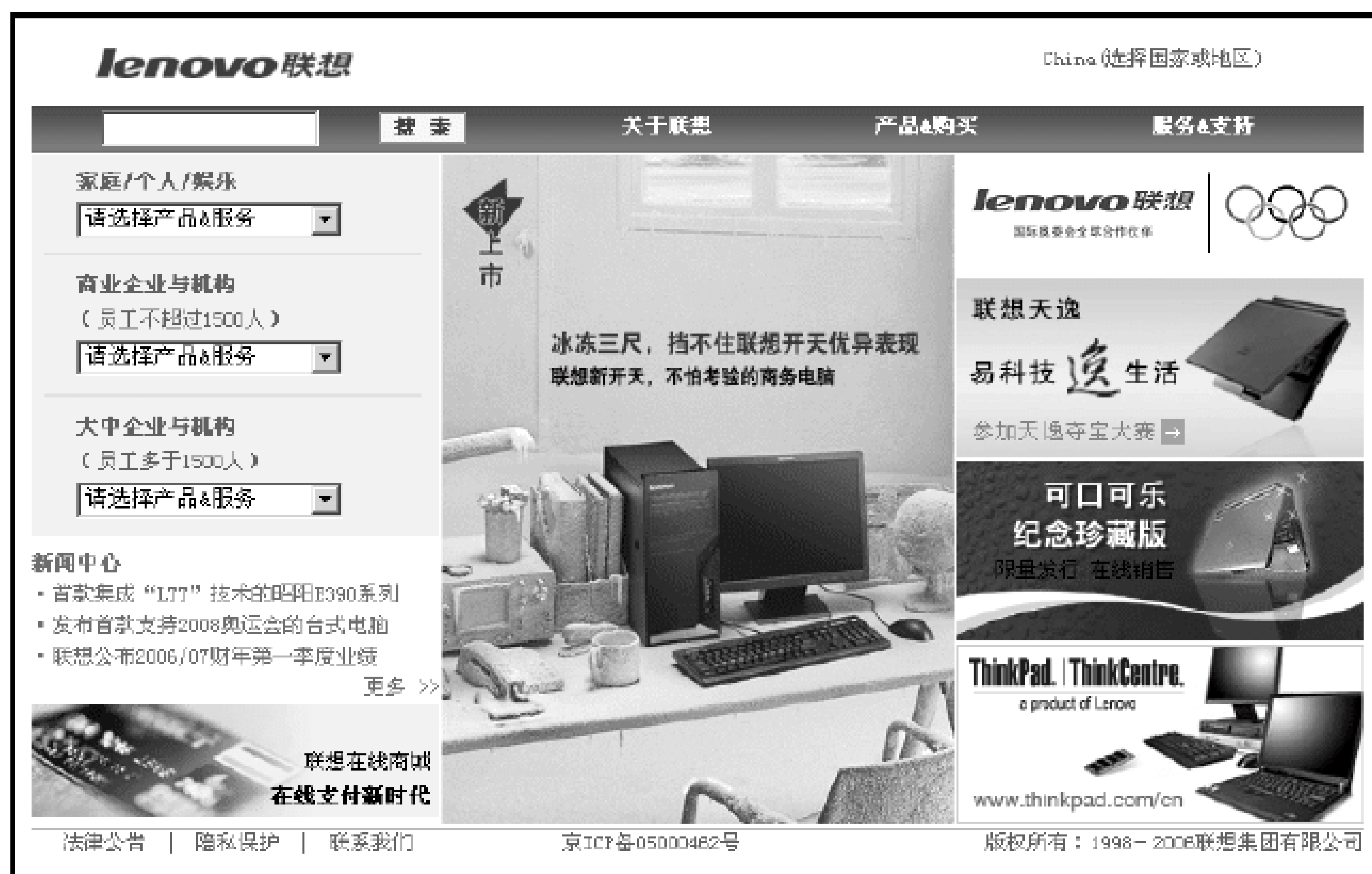


图 22.3　联想网站

关键技术

设计这类网站，关键是网页配色。针对其行业特点，建议多运用蓝色。蓝色的应用范围很广，在下面的实例中也会讲到。蓝色应用在硬件产品网站设计中，可使浏览者产生“可信”的心理。冷色的蓝还可以表现一种清爽、明快的感觉，而明度和纯度低的蓝色则可以表现一种稳重、理性的情感诉求。蓝色可以用于很多类型的网站，如计算机、企业、政府、科技等。

注意： 在建立不同类型的网站时，要用适合表达其网站特点的色彩设计。

设计过程

（1）确定网站的整体风格

硬件产品网在确立网站的整体风格时，要充分考虑其企业的经营理念和品牌形象。仔细阅读企业提供的资料及行业说明，以企业产品信息种类的多样化、行业特点的可信度等来体现整体风格。如联想网站的整体风格就是以稳重、诚信、大气、理性、简洁为主。

（2）确定页面的框架结构

页面的构架是为了更好地安排内容，在设计时要区别其主次。硬件产品属于高科技产品，种类繁多，这就要求框架设计一定要具有很好的灵活性、可视性和可操作性，整个网站看起来才不会死板。如联想网站在框架结构设计上没有采用华丽的装饰，简单明了的三分栏结构形式足以适应其信息内容，又体现了整体风格。

（3）网页色彩的运用

在硬件产品网站中，消费群体容易受到环境、文化、传统、色彩喜好的影响。在设计网站的主体色调时，建议多采用明亮的色调，如蓝色、白色等，以使整个网站显得大气，提高企业的可信度。另外，企业标准色影响着其色彩运用的定位。如联想网站就是以企业标准色——蓝色为主体色调，简单明快，减少了视觉感官刺激，符合其行业特点。

秘笈心法

心法领悟 555：建议计算机硬件网站多用对比色调。

计算机硬件产业的发展日新月异，这就要求在硬件网站设计中一定要体现出欣欣向荣、健康向上、不断进取的产业气象。另外，出于对其行业特点的考虑，建议多使用现代感强烈、能够吸引年轻人的网站设计风格，如强烈对比的黑白双色来造就冷酷、技术感强烈的版面效果。

实例 556	软件产品网 光盘位置：光盘\MR\22\556	初级 实用指数：★★

实例说明

以介绍软件为主的企业网站很多，其目的基本上都是为了发布自己企业的信息、产品信息，增强各企业之间的联系或是提高企业的品牌形象等，其设计风格大都是以诚信、庄重、灵活、大气为主。如 Sun 软件网站就是一个很好的实例，如图 22.4 所示。



图 22.4　Sun 软件网站

关键技术

细节设计在此类网站设计中很重要。一个网页设计得是否有“深度”，是否饱满，在很大程度上取决于细节设计。如果要把特定类型网站的行业特点表达出来，必须要强调一些细节。页面上的每一部分内容的放置位置、大小、颜色等都需要仔细地思考，精雕细琢，细致到每一行字（当然，这必须符合网站的整体风格）。多字或少字，页面效果就会不一样。在细节设计上处理得到位，才能将网站的整体风格更好地体现出来。

注意：在处理细节时，要充分考虑网站的整体风格。

设计过程

（1）确定网站的整体风格

一般情况下，大型软件类网站的规模较大，企业的经营理念、形象推广、企业自身经营业务便成为确立网站整体风格的出发点。网站风格通常与网站背后的传统业务有着紧密的联系。该类网站信息量大、结构复杂，为了使浏览者能够快速找到重要信息，使阅读变得轻松、简单，在整体页面布局上应体现简练、大气、稳重、灵活、富有智慧的整体设计风格，以突出其特有的企业文化和特质。

（2）确定页面的框架结构

大型软件类网站的信息内容多、更新速度快，其产品图片和广告也相对较多。横分栏式结构设计能够使页

面看起来整洁、舒适，层次感强，同时还可以制造出更多灵活变化的空间，使广告信息、产品内容信息、图片信息相互紧密结合。如 Sun 网站就具有上述特点，整体结构设计严谨、简练、大气。

（3）网页色彩的运用

软件类产品与科技相挂钩，与网站背后的传统业务有着紧密的联系，其整体经营理念和其标志标准色影响着该类网站的色彩运用。所以软件类产品在网页的色彩运用上，多数采用与企业自身形象识别相统一的色调。如 Sun 软件网站就是一个很好的实例，突出了其希望、科技、知识、大气的整体网站文化特质和氛围。

秘笈心法

心法领悟 556：设计软件产品网站要考虑的问题。

设计软件产品网站时，首先要考虑的便是广大用户的观感、软件主要的使用人群等。此外，任何软件产品都需要强大的服务支持。在软件产品网站设计时，需要重点强调服务支持的力度、方式，从而为用户提供稳定的软件环境、技术支持、软件更新维护等服务。

实例 557	物流网 光盘位置：光盘\MR\22\557	初级 实用指数：★★

实例说明

物流网站在互联网中比较特殊，它的建立是以服务客户、宣传自身企业形象为基本出发点，其网站风格大多趋向于灵活、简洁、易用。如杭州旋风货运有限公司网站在结构上比较简单，以直观的 Flash 动画为网站的主要视觉对象，突出了其行业特点和良好的视觉效果，如图 22.5 所示。



图 22.5　旋风物流网站

关键技术

此类网站的设计，关键在于 Banner 的合理运用。Banner 是整个网页设计中不可缺少的重要组成部分，能够直观地表达主题内容、宣传企业自身形象。Banner 可以是静态的，也可以是动态的，只要运用恰当，都可以将网站的氛围烘托出来。如杭州旋风货运有限公司网站就是运用动态的 Flash 动画，将整个网站的主题直观地表现出来。

设计过程

（1）确定网站的整体风格

物流属于服务性行业，主要是为客户提供便利、快捷的服务。作为客户来讲，首先想到的便是物流网站提供的服务是否诚信？企业是否正规？作为企业本身来讲，则会要求其网站能够宣传自身企业形象，从而建立一个长久的品牌形象。综合考虑各种因素，物流类网站在整体布局、色调上所形成的整体风格应该是简练、实用、美观、诚信、大气。

（2）确定页面的框架结构

从物流属于服务性行业这一特点来讲，以直观、灵活、实用的框架结构形式来进行设计可以更好地引导浏览者快速找到所需服务，同时还可以用最直接、最简单的表达方式突出其企业品牌形象。分栏式的框架结构具有很好的灵活性和可视性，所以物流网站运用分栏式的框架结构比较适合。

（3）网页色彩的运用

物流企业的行业特征决定着其色彩的运用，可以采用蓝色作为网站的主体色调。蓝色可以使浏览者产生可信的心理，很多企业在设计网站时都选择了蓝色。

秘笈心法

心法领悟 557：物流网站设计注意事项。

物流网站在整个网站的设计和风格上需要注重树立企业形象，如快捷、稳定、负责。使用配色方案时，可以将产品色与配色方案有机结合起来，这样有助于树立企业及产品形象。

实例 558 宾馆酒店网

光盘位置：光盘\MR\22\558

初级

实用指数：★★

实例说明

如今，网上预订客房已成为外出旅游、宴请宾朋最省时、省力的理想渠道之一。由于宾馆、酒店属于服务性行业，故建议此类网站采用大气、温馨、实用的网站框架，配以热情、温暖的色系。这也是此类网站在设计时的首选方式，可以显著提高吸引力。如金海棠大酒店网站（如图 22.6 所示）就是一个很不错的例子，值得借鉴和学习。



图 22.6　金海棠大酒店网站

关键技术

此类网站的设计，关键在于通过主题文字确定网页主体色调。确定网页主体色调的方法很多，运用网站本身的主题名称来确定网站的主体色调也是网页设计师经常运用的方法之一。金海棠大酒店网站就是一个比较典型的例子。由于宾馆、酒店需要营造的气氛是温馨、热情的，并且整个网站的命名“金海棠大酒店”中带有一个“金”字，这就自然而然地联想到暖色系的橙色，而橙色又很适合服务性行业，从而确立了整个网站的主体色调。

注意：主体色调的确立需要考虑的因素不是单一的，心理、民俗、传统、地域等因素都影响着整个网站主体色调的确立。在确立主体色调时应该考虑其方方面面，这样才能设计出一个优秀的网站。

（1）确定网站的整体风格

网上预订客房逐渐成为广大消费者信赖的一种消费方式。考虑其消费群体与行业本身所具有的特点，在其诉求风格上大多以情感为主要诉求点，营造一种温馨、热情、舒适的感觉，这是宾馆酒店类网站的设计方向。在框架结构上可采用简单、直观、无规律式的框架结构（无规律式的框架结构在后面将讲到），有利于浏览者快速、直观地进行浏览、预订，进而形成大气、易用、温暖、热情的网站整体风格。

（2）确定页面的框架结构

宾馆、酒店属于服务性行业，为了更好地服务于广大消费者，要求其网站能够及时更新大量信息和图片。简单、灵活、可操作性强的页面框架比较适用于此类网站。从网站的长期发展考虑，框架结构清晰、简单、大气、灵活是首选。金海棠大酒店就是最好的案例。

（3）网页色彩的运用

色彩是体现网站风格的视觉要素之一。暖色调中的橙色可以更合理地烘托网站的信息内容，更宜与网站的整体风格形成统一，可以让人联想到温暖、健康、欢喜、安全、热情，刺激作用不大，是人们普遍喜爱的色彩，适合浏览者浏览。

秘笈心法

心法领悟 558：借鉴知名酒店网站。

在酒店网站开发中，可以借鉴喜来登、海德等国际性酒店网站设计，这些酒店的网站设计较为成熟而且将各要素结合得非常和谐。

22.2　电子商务类

电子商务网站是实现消费者网上购物、商户之间的网上交易和在线电子支付的一种新型的商业运营模式，极大地方便了客户进行各种事务活动和贸易活动。其形式多变，操作方式也不相同。其设计是自由的，除了保证网站的易用性、符合经营者的理念、消费群体的喜好之外，可以任意发挥。下面通过几个实例详细介绍电子商务类网站在设计时的方法和技巧，供大家参考。

实例 559	B2C 电子商务网 光盘位置：光盘\MR\22\559	初级 实用指数：★★

实例说明

茗达电子商城是典型的 B2C（即商业机构对消费者的电子商务。这种形式一般以网络零售业为主）电子商务网。在框架上运用清晰、引导性强的二分栏结构形式，错落有致，变化灵活；在色彩上，橙黄色感觉稳重、可靠，灰色感觉高雅、大方，整个网站的色彩搭配大气，不失个性；框架与色彩的合理搭配烘托出其大气、稳重、可靠、严谨的整体风格，如图 22.7 所示。



图 22.7　茗达电子商城网站

关键技术

此类网站的设计，关键在于二分栏框架结构的应用。框架是为了合理地安排信息而设置的格局。网页框架结构应随本页面放置的信息类型与信息量等信息内容方面的相应需求而设计。在保持整体风格一致性的条件下，电子商务网可选择中规中矩的分栏结构。这样比较保险，尤其是对设计经验不太丰富的设计者来说更是这样。茗达电子商城就是采用二分栏结构设计，根据网站信息内容的划分，有重点地突出和排列信息，阅读起来十分舒适。

注意：三分栏、四分栏等框架结构也适合这种类型的网站，设计方式应视具体情况而定。

（1）确定网站的整体风格

电子商务类网站是重视效率的网站，在设计风格上相对来说比较自由。除了保证整体构架与色彩的易用性、符合经营者的理念、消费群体的喜好以外，可以任意发挥。其设计风格大多遵循自己的经营方式来规划和设计，但大方、整洁、稳重、诚信是基本要求。

（2）确定页面的框架结构

信息储存流量大、更新快是电子商务类网站的共同特点。其框架的确立主要应考虑消费群体、信息内容和网站的整体风格三方面。茗达电子商城采用的是二分栏式框架结构，引导性强，符合其信息内容的排列，再配合图片，给人一种大气、主次分明、严谨、整洁的整体感受。

（3）网页色彩的运用

电子商务网的行业特点、消费特征和心理感受影响着网页色彩的运用。橙色系被广泛应用于各种信息类型的网站，是各类消费者普遍喜欢的一种颜色。茗达电子商城采用了稳重、明快的橙黄色图片作为整个网站的视觉中心，高雅、大方的灰色为辅助色，使整个网站的整体风格突出、个性鲜明。

秘笈心法

心法领悟 559：电子商务网要考虑消费人群的审美观念。

在制作电子商务网时，还需要仔细考虑消费人群的审美观念。在实际开发中，客户的价值取向直接决定了网站的风格。

实例 560	B2B 电子商务网 光盘位置：光盘\MR\22\560	初级 实用指数：★★

实例说明

除了 B2C，电子商务类网站中还有一种 B2B 电子商务，即企业对企业的电子商务。这类网站图片与文字较多，如果运用灵活、明晰的分栏与区域相结合的框架结构形式，配以明亮、温暖、对比强烈的色调，可以更好地表达该网站的整体风格。如阿里巴巴网站在拥有大量文字信息与图片信息的情况下，还能让人感觉杂而不乱，具有亲和力，如图 22.8 所示。



图 22.8　阿里巴巴首页

下面介绍两种对比颜色的配色技巧。在多种多样的配色方案中，运用两种对比颜色进行配色，是设计师在进行配色时常用的一种手段。色彩对比指两个以上的色彩，以空间或时间关系相比较，能比较出明确的差别时，二者间相互关系称为色彩的对比关系，即色彩对比。在本例中用到的两种对比颜色是橙色和蓝色。当橙色为主

色调时，蓝色加以辅助配合，整个页面色彩丰富却不花哨，页面用色协调，布局符合形式美。

注意：广告图片的颜色也影响着网站的整体效果，在设计这类网站时，广告图片的色调一定要与整个网站的色调相互搭配、协调统一。

设计过程

（1）确定网站的整体风格

B2B 电子商务网站是企业业务发展的最佳切入点，企业与企业之间通过互联网可以直接、畅通无阻地进行产品服务及信息的交换。在整体的设计风格上应该大气、诚信、热情，能够使消费企业在登录该网站时觉得网站可信，愿意欣赏和消费。只有做到信息内容全面、图片质量清晰、页面构架井然有序，才能形成该网站的主要风格。

（2）确定页面的框架结构

B2B 电子商务网站要根据信息内容有针对性地设计。其框架结构设计多数采用分栏式结构和区域排版结构相结合的方式。这是一种实用性很强、灵活易用的框架设计结构，符合信息空间的信息存储方式主次分明、稳定、可靠。

（3）网页的色彩运用

利用两种对比色彩进行网页色彩搭配也不失为一种好的方法，即先选定一种色彩，然后选择其对比色。B2B 电子商务网站针对的消费群体范围较广，运用橙色与蓝色相互配合，是一种比较保险的配色方法，符合消费者的喜好。

秘笈心法

心法领悟 560：关于 B2B 电子商务网设计的建议。

B2B 作为企业间的交流平台，在整体风格上必须大气、诚信、热情；整体架构必须通透；使用较为保守的配色方案比较适宜，即网站配色以沉稳为主，整体色调需要热情奔放，但不失冷静、睿智。

22.3　搜索引擎类

搜索引擎网站是为世界各地用户提供查询服务的服务性门户网站，只需通过一两次单击的简单操作即可轻松找到搜索结果，从而促进了全球信息的交流。搜索引擎分为站内搜索与互联网搜索两种。下面通过几个实例介绍搜索引擎类网站设计的方法和技巧。

实例 561	站内搜索引擎 光盘位置：光盘\MR\22\561	初级 实用指数：★

实例说明

明日搜索引擎实现的主要功能是站内搜索，可以便捷地查出用户所要查询的站内信息。此处运用通栏式的框架结构，以智慧、诚信、科技、稳重的蓝色为主色调，整体风格简洁明了、方便实用，如图 22.9 所示。



搜索引擎的设计，关键在于通栏式框架结构的运用。通栏式框架结构是较为特殊的一种框架结构（所谓通栏就是一栏），适用于文字信息内容与图片内容相对较少的网站，多用于报告、科研申报等具有教育学术性质的网站，可以充分突出其行业特性和文化氛围。明日搜索引擎就是运用通栏式框架结构去适应其文字信息内容，简单明了、直击主题。



图 22.9　明日站内搜索引擎

设计过程

（1）确定网站的整体风格

由于不同类型网站的整体风格不同，也决定了站内搜索引擎放置位置、颜色的不同。在站内进行查询时，页面内容随着输入条件的不同而不同。明日搜索引擎由于其信息内容单一，在整体风格的确立上简单明了、主题突出，清晰、细致、专业是它给用户带来的第一综合感受。

（2）确定页面的框架结构

明日搜索引擎网站是有针对性的网站，所包括的内容主要是文字性信息，这就影响着页面框架结构的设计。什么样的框架结构适合大量单一文字信息呢？此处就用到了通栏式框架结构。

（3）网页色彩的运用

明日搜索引擎网站的内容形式单一，其信息内容与科技、编程内容相符合，所以在网页色彩的运用上采用科技、稳重、大气、诚信的色调——蓝色（蓝色还可以表现一种稳重、理性的情感诉求）。

秘笈心法

心法领悟 561：搜索引擎并不是一门大众技术。

搜索引擎并不是一门大众技术，从其出现开始，就是一门高门槛技术。搜索引擎实质上包括学术领域的众多先进思想和设计理念，其涉及的学科包括自然语言处理、人工智能、离散数学、排列组合、编译原理等。因此，设计一个性能良好并且实用性强的搜索引擎并不是一件容易的事。

实例 562	互联网搜索引擎 光盘位置：光盘\MR\22\562	初级 实用指数：★★

互联网搜索引擎的使命就是要为网民提供网上最好的查询服务，提供最便捷的网上信息查询方法。可以通过简洁、直观的页面构架，结合其企业形象表现其网站整体风格。如 Google 就是一个典型的实例，如图 22.10 所示。



图 22.10　Google 搜索引擎

关键技术

企业标志是企业形象的重要组成部分。企业标志通常是以图形、图像为主，当然也有一些是以文字为主的。以文字为主的标志在后面的实例中将会讲到。在各种类型的网站中，有些标志的重要性与可视性远远超过页面上的其他视觉元素。标志自身的风格对网站设计也颇有影响。标志具有强烈的形象识别功能，即使改变了颜色，网站效果也会大大不同。如在 Google 网站中，没有了标志，用户都不知道该网站的功能是什么。这就相当于一篇文章没有了主题。可见标志在网页中的重要性。

设计过程

（1）确定网站的整体风格

为了给访问者提供最快捷的网上信息查询服务，搜索引擎类门户网站必须体现其服务特点，即快速、准确。用户的目的是搜索结果而不是搜索过程，所以搜索引擎页面应该简单、品牌形象突出。可通过宣传其品牌形象、传递企业视觉信息来增强访问者的印象，让用户在下次想要查询某些资料时记住该搜索引擎网站并进行登录与查询。

（2）确定页面的框架结构

互联网搜索引擎不同于站内搜索引擎，其自身服务特点要求该网页在框架设计上易操作、直观、专业。文字信息较少，可以使页面构架设计更加自由，具有针对性。

（3）网页的色彩运用

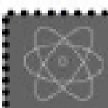
互联网搜索引擎的首页通常被称为“形象首页”。在进行网页色彩设计时，依据其自身的企业标准色以及企业形象特征即可确定该搜索引擎的主体色调，形成该网站独特的视觉效果。

心法领悟 562：简单了解搜索引擎的发展。

Google（谷歌）搜索引擎取得了巨大的成功，撼动了整个互联网世界。之后各种各样的搜索引擎服务席卷而来，从最初的 Google、Yahoo 到如今的 Baidu、MSN 等，搜索引擎的品牌越来越多，服务也越来越丰富。另外，在企业级应用的市场上，全文信息检索的需求也一直在增加，各种文档处理、内容管理软件都需要加入全文索引的功能。

22.4　生活资讯类

生活资讯类网站主要是介绍与日常生活息息相关的内容，以宣传企业形象或是介绍产品的电子商务网站居多。这类网站给用户的综合感受大都是干净、大气、温和、清新，风格独特，使人过目不忘。下面通过几个应用实例详细介绍生活资讯类网站的设计方法和配色技巧。



实例 563 家居销售网

光盘位置：光盘\MR\22\563

初级

实用指数：★★

实例说明

家居类网站与人们的生活息息相关，大部分是以产品为主的电子商务网站或是宣传自身企业的网站。“温暖”是此类网站气氛设计的关键词。例如，欧尚·宜家就是一家以家居饰品为主的生活资讯类网站，淡雅的色调、具有亲和力的家居饰品图片的运用，营造出一种祥和、舒适的居家感觉，如图 22.11 所示。



图 22.11　欧尚·宜家网站

关键技术

要实现此类网站的设计，必须充分认识到插图的重要性。图片和文字是使网页多姿多彩的主要元素，也是网站构成的基础要素。其中，插图是形成整个网站设计风格和吸引视觉注意力的重要因素。选择与网站内容相符的图片作为设计的素材至关重要，好的图片能够使浏览者一目了然地抓住网站的诉求重心，形成鲜明的视觉感受效果，从而产生愿望与欲求。

注意：插图的选取和使用不能只考虑是否漂亮，应从更多层面来考虑。总体上应该与页面的设计风格协调统一，如果不统一，就需要动手进行美化。

（1）确定网站的整体风格

确定家居类网站的整体风格时，首先想到的是其生活气息很浓郁，应该带给受众一种符合“家”这种感觉的强烈感受。利用具有亲和力的插图、舒缓而明亮的色调，可以使网站更加干净、温和、柔软，从而形成鲜明的网站风格。如欧尚·宜家就给人一种温暖、优雅、文化内涵深厚的感觉。

（2）确定页面的框架结构

家居类网站没有严格的框架设计。简单明晰、易操作、合理的框架设计既适应于信息颇多的资讯网站，又适应于宣传企业形象以及产品的网站。清晰的框架设计不仅能够形成良好的看读效果，还能够配合插图、色彩与鲜明的网站风格相统一，大气、合理。欧尚·宜家网站就是很值得学习的案例，简约时尚，卓尔不凡。

（3）网页色彩的搭配

家居类网站的主色彩最好采用明亮、干净的色调，如淡雅的米色、紫色和褐色都能够体现网站的整洁与简练，营造出生活的味道，使网站看起来与众不同，符合人们对这一行业的直观心理感受。相反色调的配色方式不宜用于家居类网站，其色彩过多、过于强烈会使画面过于跳跃，缺少柔和感。

秘笈心法

心法领悟 563：家居类网站中使用图片时的注意事项。

在家居类网站中应该大量使用图片形成层次性浏览效果，但是需要筛选图片，保持整个站点的风格不会受到图片的影响。另外，在使用插图时，需要注意尽量不要影响网站的整体风格。

实例 564	房地产信息网 光盘位置：光盘\MR\22\564	初级 实用指数：★★

实例说明

房地产行业个性化、形象化竞争日趋激烈，其自身经营理念、形象识别等因素决定着网站的风格。该类网站通常在框架结构与色彩搭配上比较自由，可以任意发挥，但是必须符合其房地产形象定位。追求创意与个性是现代房地产业的共性。如锦江实业网站就是一个具有鲜明个性的典型例子，它打破了传统框架与色彩的网站形式，运用个性化的导航、图像艺术充分体现出深厚的企业文化内涵，值得读者学习，如图 22.12 所示。



图 22.12　锦江实业网站

此类网站的设计关键在于图像艺术。网页设计离不开图像，图像艺术更多的是靠创意。一个好的创意可以形成网站的风格，产生良好的看读效果。但是有些图像素材在进行设计时不是很完美，这就需要围绕其创意，利用软件的功能将不完美的图像素材制作成符合创意的图像。有时为了达到完美的创意效果，还需要将多张图片进行组合。如锦江实业右上角的图片就是经过修饰组合后所显示的效果，流畅、洒脱、主题突出，形成整个网站的视觉中心。

设计过程

（1）确定网站的整体风格

房地产行业通常运用开发项目的定位、建筑设计的理念、策划方案的创意来确定其品牌形象，再通过品牌形象识别确定网站的主体色调与框架结构等。追求个性化、形象化已经成为现代房地产行业的主要特点。在确立该类网站整体风格时，除了多考虑一些品牌形象因素之外，还应该考虑与传统媒体相接轨，从而达到统一的宣传效果。如锦江实业网站的整体风格大气、形象突出，值得学习。

（2）确定页面的框架结构

房地产类网站的框架结构通常根据其品牌定位而确立，层次分明、重点突出，与整体风格相统一是在设计此类网站框架结构时应该注意的问题。

（3）网页色彩的运用

房地产类网站在色彩运用上最好采用明亮、个性、符合其行业特质和品牌形象的色调。代表沉着、安定、古香古色、富贵等的色调是比较个性的色彩形式，也通常被用于房地产类网站。这种颜色通常是运用纯色加黑加以调出。

秘笈心法

心法领悟 564：设计房地产网站时要注重房产信息的时效性。

房地产网站提供了各个地区的房源信息，必须保证这些信息的时效性，为用户提供多方面的信息，另外对于网站中的房源信息的搜索也要做到位、做到功能强大。

22.5　娱乐类网站

提到娱乐类网站，读者可能会联想到五颜六色的色彩、CD 里的悠扬歌曲、经典的 Flash 动画、夸张的游戏画面以及明星新闻。这类网站中的插图夸张、文字内容丰富、色彩明亮、娱乐气氛浓厚。个性化的表现手法也经常运用于该类网站中。下面通过几个应用实例介绍娱乐类网站在设计与配色上的技巧和方法，供大家参考。

实例 565	音乐网 光盘位置：光盘\MR\22\565	初级 实用指数：★★

实例说明

不论是以音乐信息内容为主还是以个人音乐为主的网站，都属于音乐网站。音乐网站的风格特点仍然是以其自身的文化特点为设计依据，如以介绍音乐信息内容为主的音乐网站风格大多时尚、热情、色彩丰富、结构灵活，而太过个性化的设计手法只适用于以宣传歌手、俱乐部等为主的音乐网站。比较有名的 TOM 音乐网主要是以宣传音乐信息为主，其设计色彩丰富、时尚、热情，结构灵活，信息全面，在浏览时感觉非常舒适，如图 22.13 所示。



此类网站的设计，关键在于横式的分栏式框架结构。前文讲到的分栏式框架结构都是以竖分栏为主的框架形式，而在此所要介绍的是与其相对应的横分栏框架结构。运用横分栏进行页面布局与导航，可以使大量的信息内容更好地排列，为整个页面节省更多的空间，使整个页面结构更清晰，内容的条理性更强。TOM 音乐网站就是以横式的分栏式结构进行组织、排列其需要宣传的信息内容的，在结构运用上灵活、严谨。

注意：无论运用竖式的还是横式的框架结构，必须依据其信息内容、网站风格而定。



图 22.13　TOM 音乐网站

设计过程

（1）确定网站的整体风格

以宣传信息内容为主的音乐网站所针对的消费群体相对比较集中，以中青年人居多。这些访问者有的是为了了解音乐人以及音乐专辑的最新信息，有的是为了下载最新的流行音乐。由于其行业特点是以娱乐为主，这就需要在设计这类网站时突出其专业、整洁、热情、内容全新的设计风格，以此来达到宣传的目的。

（2）确定页面的框架结构

横式分栏框架结构形式可以适应音乐网站内容更新快、信息结构复杂、导航个数多等特点，节省更多的页面空间，更好地排列其大量的信息内容。如 TOM 音乐网就是音乐网站的代表。

（3）网页色彩的运用

音乐网站在色彩运用上，以热情、时尚的色调为主。如 TOM 音乐网站就是以比较热情、保险、温暖的橙色为主色调，让用户在浏览该网站时感受到强烈的音乐氛围，更好地寻找自己要查询的内容。

秘笈心法

心法领悟 565：开发一个 MP3 在线音乐网。

MP3 音乐在线播放在网络上比较流行，如百度 MP3 等，用户可以随时收听自己喜欢的歌曲来放松心情。读者不妨试着开发一个 MP3 在线音乐网站，并提供给用户对某一首歌曲进行单一的播放和歌词同步显示的试听功能、选择自己喜欢的多首歌曲进行顺序播放、随机播放和单曲播放的歌曲播放功能以及下载功能，供用户将自己喜欢的歌曲下载到本地进行收听。

实例 566	电影网 光盘位置：光盘\MR\22\566	初级 实用指数：★★

电影网属于娱乐类网站中的一种类型，考虑其行业特点，在设计此类网站时需要营造出网站的电影文化氛围。如图 22.14 所示的电影网以黑色为主色调，表现出一种神秘、稳重、诡异的气氛，再与影片插图相统一，充分体现出其影片情感，将电影所具有的原味文化表现得淋漓尽致。



图 22.14　中艺电影网站

关键技术

如图 22.14 所示电影网的设计，关键在于黑色的运用。黑色是一种很特殊的色彩，为全色相，没有纯度，给人以黑暗、恐怖、刚正、忠毅、神秘、坚硬之感。本身无刺激性，但是与其他色配合能增加刺激，并能取得很好的效果。如图 22.14 所示的电影网就是运用黑色来营造一种神秘、恐怖的电影文化，非常具有典型性。

设计过程

（1）确定网站的整体风格

电影网娱乐性强，对比强烈、主题明确是表现其风格的基础。电影本身所蕴涵的现代感和文化特质，是确定网站风格的重要依据。因此，此类网站应该体现出大气、主题鲜明、现代感强、视觉冲击感强烈的特点。

（2）确定页面的框架结构

电影网通常是以介绍电影信息内容或者宣传某部影片为主，所以在页面的构架设计上多采用简单、灵活、清晰感较强的区域式框架结构设计。通过将页面中的信息内容划分成块，可以更好、更灵活地引导消费者，还可以适应多种信息内容编排的需求，使浏览者在浏览时能够阅读舒适且快速找到自己所需内容。

注意： *在运用区域式框架结构设计时，并不是“教条”地选择，而是要配合信息内容有针对性地设计。信息内容形式影响着框架设计。读者在学习时应该灵活运用。*

（3）网页色彩的运用

为了突出文化特质，黑色作为一种无彩色，越来越多地被设计师用于电影网。无彩色与彩色进行配色是常用的配色方式。黑色可以稳住跳跃的色彩，使画面和谐，利于达到统一的视觉效果，同时还能给人以稳重的心理感受。如图 22.14 所示电影网就是运用黑色为主体色调，配合影片插图，使整个网站的氛围与整体感觉非常强烈。

心法领悟 566：在电影网中可提供大量 Flash。

在电影网制作中，可以考虑大量使用 Flash 播放影片的预告片、片花，为客户提供更好的体验度。另外，对于这类网站，建议使用浅色系作为主配色，但是在点睛色的选择上可以考虑使用补色等对比度较为强烈的颜色。

实例 567

游戏门户网

光盘位置：光盘\MR\22\567

初级

实用指数：★★

实例说明

作为游戏类网站中的一种分类，拥有庞大的信息量和用户资源是游戏门户网站的优势所在。该类网站具有强大的娱乐性，网站设计整体风格大气、有个性。例如，明日游戏门户网站以突出的色彩设计给浏览者留下深刻的印象。从整体上看，网站色彩多变、色调统一，形成了统一的色彩风格，充分体现出娱乐网站的气氛，如图 22.15 所示。



图 22.15　明日游戏网站

关键技术

如图 22.15 所示为游戏门户网站的设计，关键在于红色的运用。红色给人的具体联想是火焰、太阳、血，象征着热烈、危险、活力、愤怒。红色的纯度高，注目性高，刺激作用大，能增高血压、加速血液循环。用红色为主色的网站并不多，在大量信息的页面里如有大面积的红色将不利于阅读。但是如果搭配合理，则可以体现出一种振奋人心、先进、野性、神秘的感觉。明日游戏门户网站就是运用红色为主色调，暗红、橙色、黑色相配合，烘托出一种神秘、刺激、热烈、活力的味道。

设计过程

（1）确定网站的整体风格

游戏类网站由于本身具有强大的娱乐性，具体的游戏网站应以其文化特点作为设计的切入点。游戏门户网站的内容比较丰富，易用、易读、易查是该类网站应该具备的基本特点。合理地运用游戏插图来烘托网站的娱乐性，也是在设计该类网站时所采用的一种好方法。它可以烘托出娱乐性这一特点，从而形成大气、个性、整洁的页面设计，既符合门户类网站的特点，又符合娱乐类网站的特点。

（2）确定页面的框架结构

由于游戏门户网站具有信息量大、用户资源广的特点，采用分栏式与区域式的框架结构，可以使浏览者在浏览时操作方便、引导性强、简练、清晰、易读、易查、易用。这种可操作方式可以适应大量信息内容编排的需求。明日游戏门户网站就是运用区域式框架，灵活地安排其信息内容，值得读者学习。

（3）网页色彩的运用

提到游戏网站，必定让人联想到夸张的造型、神秘迷幻的暗深色调的配色，但这不是绝对的。游戏本身的特性也决定着其网页色彩。如明日游戏门户网站就是运用黑色和红色的配色方式来烘托网站神秘、刺激的整体感觉。

必法领悟 567：充分运用游戏网站中的游戏插图。

在开发过程中，可充分运用游戏网站中游戏插图的特点来设计不同感觉的游戏网站，如轻松明亮的、神秘迷幻的。选用不同的图片，风格也会不同。

22.6　供求信息类

供求信息类网站与日常生活息息相关，是为用户提供一定生活服务的网站，可以起到人与人之间进行沟通的桥梁作用。其分类很广，这里着重介绍在设计人才供求和商品供求的网站时色彩运用的方法和技术。

实例 568	人才供求网 光盘位置：光盘\MR\22\568	初级 实用指数：★★

实例说明

人才供求网作为供求信息网站的一种，主要是为众多的招聘单位与求职人员提供一个信息化服务平台。基于该类网站的行业特征等因素，要求此类网站在风格设计上应做到诚信、大气、专业、全面。如智联招聘网就已经形成了自己的品牌，是一个典型的成功实例，如图 22.16 所示。



图 22.16　智联招聘网

关键技术

此类网站的设计，关键在于导航与网站风格的统一。导航与网站风格有着密不可分的关系，浏览者在网站中会本能地寻找导航，导航是网站信息内部的入口，其风格决定着网站风格；反之，网站风格也可以决定导航风格。二者相互呼应，页面才可以形成统一的风格。智联招聘网站导航条的运用就与网站的整体风格有着不可分割的密切关系。

注意： 此处还要提到的一点是，主导航与从导航这两者之间的关系处理的好坏，对网站的整体风格也有影响。

（1）确定网站的整体风格

人才供求网站针对的浏览者特征比较明显，这就要求其整体风格必须体现出诚信、大气、安全、全面等。

这样的网站可以让受众人群在充分信任该网站的基础上感受其信息的最全与最新，愿意在该网站发布信息。框架设计灵活简练，色调设计体现出诚信，整个页面干净、整洁、大气，才能吸引浏览者的目光。

（2）确定页面的框架结构

人才供求网站的整体风格一旦确立，框架结构设计就要充分将其风格表现出来。分栏式的框架结构形式比较适合人才供求网站。人才供求网站以提供信息为主要服务内容，信息才是该类网站的重点，而层次分明、主题突出的框架结构才能更好地让浏览者找到所需内容。再利用文字的精致排列与插图进行相互配合，可以使阅读更舒适，注目性更高。

（3）网页色彩的运用

为了使人才供求网站本身的行业特征更加明显，运用蓝色这一色系的颜色来进行搭配是比较保险的配色方案，可以充分体现出人才供求的诚信、安全、大气。如智联招聘网就运用蓝色为网站的主体色彩，与标志中的蓝色进行统一，再运用蓝色的对比色——橙色进行辅助搭配，产生良好的视觉效果。

秘笈心法

心法领悟 568：尝试运用标志的标准色。

在设计过程中，可尝试着运用标志的标准色来确定网站的主体色调。在设计各种风格的网站时，只有协调好导航与网站整体风格的关系，才能设计出更精彩的网页。

实例 569　二手商品供求网

初级

光盘位置：光盘\MR\22\569

实用指数：★★

除了人才供求之外，还有其他多种类型的供求网站。例如，以二手商品供求为主的网站如今受到越来越多网民的喜爱，将人们的生活紧密地联系在一起。如 IT 导购助手网站就是以二手 IT 产品为主的网站，其信息内容全、新，风格明晰、诚信、整洁、灵活，将 IT 业的行业特点充分地表现了出来，如图 22.17 所示。



图 22.17　IT 导购助手网站

关键技术

此类网站的设计，关键在于实现网站框架结构的易用性。前文曾用大量的篇幅叙述了网站框架结构的分类以及在网站中的应用，在此还要强调一下框架结构的易用性。易用性原则是自始至终贯穿整个网站设计过程的要点，使站点易于使用，是从设计网站框架结构开始就应该放在第一位的首要问题。尤其是针对应用性较强的电子商务网站、门户网站和资讯网站等，应将常用的信息模块放在页面的上部，并在格局上多多考虑如何突出重点信息，以便浏览者可以快速地跳转到其他信息分层中。

注意： 除了框架结构的易用性，还有其他设计环节的易用性问题，在设计网站时应该细致、全面地进行考虑。

设计过程

（1）确定网站的整体风格

二手商品供求网站的创建目的是为了给消费者提供一个买卖二手物品的交易平台。其整体氛围是急躁的，消费者多数没有闲情雅致去慢慢地欣赏网页，突出其商品信息是此类网站设计应该重点把握的问题。采用明亮的色调、简练明晰的框架结构、简单直白的产品插图是设计该类网站时通常运用的方法，可以充分地表达出网站整洁、诚信、灵活的风格特点。如 IT 导购助手网站就是一个典型的以二手 IT 商品为主的供求信息网，它充分运用对比色的配色方案将网站的风格表现得淋漓尽致。

（2）确定网页的框架结构

由于二手商品供求网站是以经营二手商品为主要目的，其商品信息、产品图片等信息繁多。这就要求其网页在框架结构上必须做到灵活、易操作，以此来适应其大量的信息内容。分栏式框架结构用在该类网站上是最合适的。它可以制造出更多灵活变化的空间，使商品信息与产品图片更好地进行陈列，从而形成网站的整体风格。例如，IT 导购助手网站就符合以上所述特点。

（3）网页色彩的运用

由于二手商品供求网站以宣传产品和供求信息为主，这一行业特点要求该类网站不宜运用大幅的图片和大面积的色块。可运用干净、明快、整洁的颜色为主体色，以达到适宜阅读、浏览舒适的效果。如 IT 导购助手网站就运用干净、明快的浅橙与浅蓝进行对比色配色，取得了良好的视觉效果。

秘笈心法

心法领悟 569：供求信息网站如何吸引用户。

对于供求信息网站来说，用户的访问量是至关重要的。如果网站的访问量很低，则很难吸引企业，提供不了有偿服务，也就没有利润可言了。因此，供求信息网站必须提供大量的、免费的、有价值的信息才能够吸引越来越多的用户。为此，网站不仅要为企业提供各种有偿服务，还需要额外为用户提供大量的无偿服务。

22.7 其 他 应 用

前面讲解的网站分类方式并不是很严谨，换句话说，没有一种严谨的方式可以将网站类型加以划分。这里主要介绍前面实例中没有讲到的、具有一定代表意义的网站类型，如个人主页、美食、论坛、博客等。

实例 570	个人主页 光盘位置：光盘\MR\22\570	初级 实用指数：★★

实例说明

在互联网上，千奇百态、风格各异的个人主页是商业网站所不能比拟的。这些网站在网站制作技术、创作

范围、色彩运用上没有严格限制。信息结构决定着个人网站的结构及风格创意。如崔宏远的个人主页，无论是在制作技术上，还是创作风格上都比较有个性，充分展示出自己独特的风格，如图 22.18 所示。



图 22.18　崔宏远个人网站

关键技术

此类网站的设计，关键在于 Flash 技术的应用。大量使用 Flash 对于信息量庞大的网站来说并不适宜，因为它将直接影响到网站的浏览速度。个人网站信息量不大，在 Flash 的运用上给设计师带来了很大的发挥空间。Flash 特殊的制作手法将网站独特的风格演绎得淋漓尽致，技术含量高、时尚、个性、现代、动感十足。同时 Flash 技术也越来越多地被运用到网站功能设计上，如使用 Flash 进行导航等。

注意： Flash 技术不能随意滥用。网站设计并不是“动就是美”，太过个性化的 Flash 技术不适合用在信息量大的网站上。

设计过程

（1）确定网站的整体风格

个人主页是互联网上一道绚丽的风景。其风格的确定是由设计者的设计初衷和信息结构决定的，而制作技术、个人喜好的不同也影响着网站的整体风格。例如，有的是为了宣泄个人情感；有的是为了让别人了解自己；有的是为了和朋友们进行交流……很明显，个人网站的整体风格自由，只要能表达出自己的创作意图，能以独特气质和个性来吸引浏览者就是成功的作品。

（2）确定页面的框架结构

根据个人网站整体风格的定位，个性是个人网站所要表达的主题。个人网站信息内容相对较少，在确立该类网站的构架上要根据其内容信息进行设计。分栏结构、无规律框架等各式各样的框架结构都适合个人网站。

（3）网页色彩的运用

个人网站创作意图不同，色彩运用也是随意、自由的。基本上，网站内容和网站气氛决定了网站用色。厚重的色彩，可以展现出稳重、执着的创作意图；而淡雅的色调可以给人一种温和、富有格调的感觉。

秘笈心法

心法领悟 570：个人空间网站。

个人空间也称为个人主页，即一个属于自己的地盘。这里开发的个人空间网站只是一个小型项目，主要目的是演示如何实现个人空间的创建。在个人空间创建过程中，可以命名个性化空间名、上传个性化头像、展现个性签名等。用户在创建个人空间前，需首先注册为会员并进行登录。首次登录后将会创建个人空间，以后再次登录时就将直接进入到个人空间管理首页。在个人空间管理首页中，可以发表最新个人日志，并可对发表的日志进行管理（修改和删除等操作），与好友分享喜怒哀乐。

实例 571　美食网　初级

光盘位置：光盘\MR\22\571　实用指数：★★

实例说明

随着经济的发展，餐饮业的个性化消费日趋明显，上网了解、沟通、学习不同地域的美食特点和制作方法已经成为一种时尚。这类网站在设计时通常以情感为诉求点，强调网站的健康、绿色、热情、温暖。如天津美食网强调了美食的健康与绿色，让更多用户可以更方便地了解天津的餐饮市场和餐饮特点，如图 22.19 所示。



图 22.19　天津美食网

关键技术

此类网站的设计，关键在于认识字体标志的重要性。前面曾介绍过标志的重要性，在这里强调一下以网站的名称作为标志的特点。文字标志看起来很简单，没有图形、图像的变化，而是直接运用主题文字来定义，比较直观、易记、易懂，有着强烈的形象识别功能。如天津美食网就是运用黑色的、比较直观的字体来做标志，具有手写字一样大气、美观的特点，将浓郁的地域文化表现了出来。

设计过程

（1）确定网站的整体风格

网上餐饮市场包罗万象，菜系繁多，充分满足了消费者学习、沟通、了解美食特点和制作方法的要求。对于这类网站，不同的地域文化、人文气息、菜系自身的定位是在确定网站整体风格时所必须考虑的重要因素。如四川美食在设计时要有川味；天津美食在设计时要有津味等。其风格在确定时一定要体现出温暖、健康、绿色、热情。天津美食网的地域文化特征等表现得很到位，值得学习。

（2）确定页面的框架结构

在美食类网站中，令人垂涎欲滴的美食图片和大量的文字信息是必不可少的，它们是网站构成的基础要素。其更新速度很快，这就要求该类网站的框架结构一定要做到实用性强、灵活易用，这样才能适应图片与文字的经常变化。建议采用分栏式框架结构，因为它符合信息存储灵活的页面架构方式，这样既不影响网站的整体风格，也不影响大量信息的存储，一举两得。

（3）网页色彩的运用

餐饮市场消费的个性化已经成为市场的一大特点，突出健康美食和绿色餐饮已成为行业的重要选择。天津美食网以绿色为主、黄色为辅的配色方式，充分营造出健康、绿色、热情、温暖的风格特点。

秘笈心法

心法领悟 571：在美食网站中利用 JavaScript 控制美食图片播放。

要在网页中实现循环播放图片，可以通过 JavaScript 脚本和调用 Flash 文件两种方法来实现，而通过 JavaScript 脚本实现又可以分为网站头部的循环播放图片和循环不间断显示图片两种。当公司美工人手不足时，开发人员可以使用 JavaScript 脚本控制循环播放图片；当公司美工人手充足时，可以让美工将循环显示的图片做成 Flash，然后开发人员直接在程序中调用即可，这样可以缩短开发周期。与 HTML 语言提供的 marquee 元素相比，通过

JavaScript 脚本可以使图片循环不间断显示，这样使网页看起来更美观。

实例 572	博客网站 光盘位置：光盘\MR\22\572	初级 实用指数：★★

实例说明

博客（Blog）是一种网络交流方式，为广大网民提供了一个信息发布、知识交流的传播平台。一个 Blog 就是一个网页。它与 BBS 的区别在于，BBS 开放性、自由性、随意性强，而 Blog 的内容是经过使用者的思考和精心筛选组织起来的。如明日博客网站定位明确，清新、整齐、简洁、易用、个性化十足，秉承了个人网站的自由风格，如图 22.20 所示。



图 22.20　明日博客网站

关键技术

此类网站的设计，关键在于网站的气韵与风格设计。任何设计合理、优美的网站都会形成一种特有的气质氛围，给用户留下深刻的印象。这种感觉就是风格。风格的确定是从理性到感性的过程。首先找出其行业特性；其次分析网站内容，从信息类型、信息量以及信息本身的需求出发，找出网站的表达方式（如信息的结构布局、导航条数量和形式等），以此向浏览者渗透网站文化理念，在其心中树立网站独特的形象。

注意：风格的确定需要把现有若干个因素考虑在内，将若干个可设计环节连成面，相互配合、相互呼应、重复统一、保持一致，而不是依靠单独的物体。

设计过程

（1）确定网站的整体风格

博客一个私有性较强的平台，面向的是个人和较小的、具有共同目标的群组，是服务于个人和小团体的。当今社会凸显个人才能、张扬个性、服务于特定对象的需求日益突出，这就影响着博客网站的自身定位，决定着网页的风格。该类网站的设计风格偏向于清新、整齐、简洁、易用，用户定位明确，从而形成其风格的鲜明性。

（2）确定页面的框架结构

由于博客类网站秉承自由精神，设计风格相对比较自由、个性，所以在确定页面的构架上也没有固定的模式。规律式的、无规律式的框架结构都适用于该类网站，只要框架结构符合其内容信息和整体风格即可。

（3）网页色彩的运用

运用清新、典雅、温和、富有格调的色彩搭配是博客类网站常常采用的配色方式。例如，明日博客网站就是运用绿色为主体色调，体现出自由、明亮的整体风格。

心法领悟 572：简单了解博客的产生原因。

博客的个性化和平民视角使得它提供的消息更贴近人们的生活，所以很多用户都想建立自己的网络空间。传统的网络交往方式主要是留言本、BBS（论坛）、聊天室及 IM（即时通信）等，但它们或多或少都存在不足。留言本主要用来留言，不能进行留言回复；BBS 主要用来探讨问题；IM 要想发挥作用，必须要求交流的双方同时在线；而聊天室更是闲人的乐园。博客的存在，可以说是一种网络的虚拟社区。在这里用户可以通过网络日志的形式方便、快捷地发表自己的心得体会，及时、有效并轻松地与他人交流。

第23章 Java Web典型项目开发案例

- Ajax 聊天室
- 博客网核心模块开发
- 在线投票统计功能
- B2C 电子商务网站
- 在线音乐
- 校内数码相册
- 仿百度知道之明日知道

23.1 Ajax 聊天室

实例 573　实时获取并显示在线人员列表　初级

光盘位置：光盘\MR\23\573　　实用指数：★★★★

实例说明

在开发聊天室程序时，为了让用户及时了解在线用户，并与其他用户进行交流，需要提供实时获取并显示在线人员列表的功能。本实例将介绍如何通过 Ajax 实现实时获取并显示在线人员列表，运行结果如图 23.1 所示。

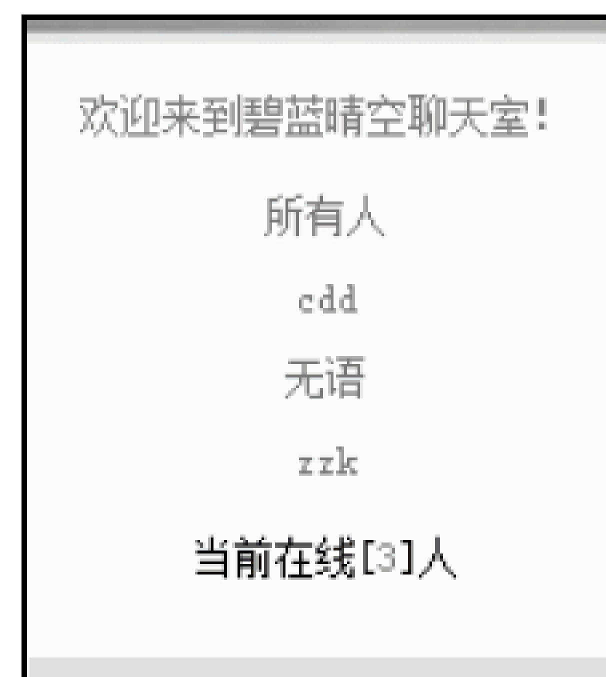
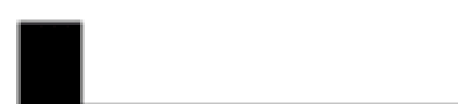


图 23.1　实时获取并显示在线人员列表



本实例主要应用 Ajax 重构技术来实现。随着 Ajax 应用程序的不断扩展，越来越多的 JavaScript 代码被应用到 Ajax 中，这可能会导致许多意想不到的问题，因此有必要对 Ajax 代码进行重构。下面将介绍实现 Ajax 重构的基本步骤。

（1）创建一个单独的 JS 文件，名称为 AjaxRequest.js，并且在该文件中编写重构 Ajax 所需的代码。具体代码如下：

```
var net=new Object();                //定义一个全局变量 net
//编写构造函数
net.AjaxRequest=function(url,onload,onerror,method,params){
  this.req=null;
  this.onload=onload;
  this.onerror=(onerror) ? onerror : this.defaultError;
  this.loadDate(url,method,params);
}
//编写用于初始化 XMLHttpRequest 对象并指定处理函数，最后发送 HTTP 请求的方法
net.AjaxRequest.prototype.loadDate=function(url,method,params){
  if (!method){
    method="GET";
  }
  if (window.XMLHttpRequest){
    this.req=new XMLHttpRequest();
  } else if (window.ActiveXObject){
    this.req=new ActiveXObject("Microsoft.XMLHTTP");
  }
  if (this.req){
    try{
      var loader=this;
      this.req.onreadystatechange=function(){
        net.AjaxRequest.onReadyState.call(loader);
      }

      this.req.open(method,url,true);
        if(method=="POST"){
            this.req.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
        }
      this.req.send(params);
    }catch (err){
      this.onerror.call(this);
    }
  }
}
```

```
//重构回调函数
net.AjaxRequest.onReadyState=function(){
   var req=this.req;
   var ready=req.readyState;
   if (ready==4){
      if (req.status==200 ){
         this.onload.call(this);
      }else{
         this.onerror.call(this);
      }
   }
}
//重构默认的错误处理函数
net.AjaxRequest.prototype.defaultError=function(){
   alert("错误数据\n\n 回调状态:"+this.req.readyState+"\n 状态: "+this.req.status);
}
```

（2）在需要应用 Ajax 的页面中应用以下语句包含 JS 文件 AjaxRequest.js。

```
<script language="javascript" src="JS/AjaxRequest.js"></script>
```

（3）在应用 Ajax 的页面中编写错误处理的方法、实例化 Ajax 对象的方法和回调函数。具体代码如下：

```
<script language="javascript">
/******************错误处理的方法*******************************/
        function onerror(){
                alert("很抱歉，服务器出现错误，当前窗口将关闭！");
                window.opener=null;
                window.close();
        }
/******************实例化 Ajax 对象的方法*******************************/
        function showOnline(){
                var loader=new net.AjaxRequest("online.jsp?nocache="+new Date().getTime(),deal_online,onerror,"GET");
        }
/************************回调函数*************************************/
        function deal_online(){
                online.innerHTML=this.req.responseText;
        }
</script>
```

（1）创建一个单独的 JavaScript 文件，名称为 AjaxRequest.js，具体代码参见本实例的“关键技术”。

（2）在聊天室的主界面 main.jsp 中，应用代码中包含 AjaxRequest.js 文件。

（3）编写自定义的 JavaScript 函数 showOnline()，用于实例化 Ajax 对象。showOnline()函数的具体代码如下：

```
function showOnline(){
      var loader=new net.AjaxRequest("online.jsp?nocache="+new Date().getTime(),deal_online,onerror,"GET");
}
```

在上面的代码中，一定要有?nocache="+new Date().getTime()语句，否则将出现在线人员列表不更新的情况。

（4）从上面的代码中可以看出，Ajax 异步请求目标的 URL 地址为 online.jsp，即 JSP 文件。在该文件中，主要是将保存在集合类中的在线人员列表显示到页面。online.jsp 页面的代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java" import="java.util.*" %>
<%@ page import="com.wgh.UserInfo"%>
<% request.setCharacterEncoding("gbk"); %>
<%
UserInfo list=UserInfo.getInstance();
Vector vector=list.getList();                                    //获取在线人员列表
int amount=0;
%>
<table width="100%" border="0" cellpadding="0" cellspacing="0">
   <tr><td height="32" align="center" class="word_orange ">欢迎来到碧蓝晴空聊天室！</td></tr>
   <tr>
   <td height="23" align="center"><a   href="#" onclick="set('所有人')">所有人</a></td>
   </tr>
 <%if(vector!=null&&vector.size()>0){
      String username="";
      amount=vector.size();
            for(int i=0;i<amount;i++){
```

```
            username=(String)vector.elementAt(i);                //获取在线用户的用户名
        %>
    <tr>
      <td height="23" align="center">                            //显示在线用户的用户名
          <a href="#" onclick="set('<%=username%>')"><%=username%></a></td>
    </tr>
<%}}%>
<tr><td height="30" align="center">当前在线[<font color="#FF6600"><%=amount%></font>]人</td></tr>      //显示在线人数
</table>
```

（5）在聊天室的主界面 main.jsp 中，将左侧用于显示在线人员列表的单元格的 id 属性设置为 online，用于实时显示在线人员列表。具体代码如下：

```
<td width="165" valign="top" bgcolor="#F7FDED" id="online" style="padding:5px">在线人员列表</td>
```

（6）在聊天室的主界面 main.jsp 中编写 Ajax 的回调函数 deal_online()，用于将获取的在线人员列表赋值给 id 为 online 的<td>标签的 innerHTML 属性。deal_online()函数的具体代码如下：

```
function deal_online(){
    online.innerHTML=this.req.responseText;
}
```

（7）为了让页面载入后就调用 Ajax 获取在线人员列表，并且每隔 10 秒便获取一次数据，还需要在 main.jsp 页面中添加以下 JavaScript 代码：

```
window.setInterval("showOnline();",10000);
window.onload=function(){
    showOnline();                          //当页面载入后显示在线人员列表
}
```

秘笈心法

心法领悟 573：解决应用 Ajax 进行异步请求时结果不更新的情况。

在应用 Ajax 进行异步请求时，有时会出现请求的结果不更新的情况。这是由于浏览器认为该请求已经处理完毕，所以不再向服务器重新发送请求所造成的。解决该问题的方法是，在请求的 URL 地址中添加一个时间戳?nocache="+new Date().getTime()，这样就可以保证每次的请求都是新的，从而解决了请求结果不更新的问题。

实例 574	实现用户发言	初级
	光盘位置：光盘\MR\23\574	实用指数：★★★☆

实例说明

在聊天室程序中一个必不可少的功能就是实现用户发言，也就是发表自己要说的话。为了增加聊天室的魅力，在实现用户发言时，通常需要提供让用户选择自己的表情、字体颜色、是否为悄悄话等功能。本实例将介绍如何实现用户发言，并将发言信息保存到 XML 文件中。实例运行结果如图 23.2 所示。

图 23.2　实现用户发言

本实例中应用的主要技术是 Ajax 重构和通过 JDOM 解析 XML 文件。关于 Ajax 重构的方法，参见实例 573。下面对 JDOM 及应用其解析 XML 文件所涉及的技术进行介绍。

JDOM 因其简洁易懂的 API 被广泛地使用。JDOM 提供了 org.jdom、org.jdom.input、org.jdom.output、org.jdom.adapters、org.jdom.ransform 包。其中 org.jdom 包中的类是解析 XML 文件所要用到的所有引用数据类型；org.jdom.input 和 org.jdom.output 两个包分别用来处理 XML 内容的输入、输出。下面简单介绍一下 JDOM

解析 XML 文件时用到的类。

- SAXBuilder 类：当要读取 XML 资源时，要使用 input 包下的 SAXBuilder 类从输入流构建文档对象。
- Document 类：该类代表文档对象。
- Element 类：该类代表 XML 元素。
- Comment 类：该类代表 XML 文件中的注释。

例如以下 XML 文件中的内容：

```
<?xml version="1.0" encoding="GBK"?>
<bookroom>
  <book>
      <bookname>
          Java 数据库系统开发案例精选
      </bookname>
      <author>
            王国辉等
      </author>
      <date>
        2007 年 3 月
      </date>
      <price>
            49 元
      </price>
  </book>
</bookroom>
```

上述的文档用 Document 来抽象，bookroom 是文档的根元素，用 Element 类封装，可以通过 Document 对象的 getRootElement()方法获取 Element 对象。Element 元素可以有子元素，如 bookname、author、date、price 都是子元素，可以通过 Element 对象的 getChildren()方法获取这些子元素。Text 代表了 XML 中的文本值，如元素属性值、元素值、注释的内容等。例如，上述代码中的<price>49 元</price>中元素值为“49 元”。

（1）在聊天室主界面的合适位置添加用于收集用户发言信息的表单及表单元素。关键代码如下：

```
<form action="" name="form1" method="post" >
    <input name="from" type="hidden" value="<%=session.getAttribute("username")%>"> [<%=session. GetAttribute ("username")%> ]对
    <input name="to" type="text" value="" size="35" readonly="readonly">
表情
<select name="face" class="wenbenkuang">
  <option value="无表情的">无表情的</option>
  <option value="微笑着" selected>微笑着</option>
  <option value="笑呵呵地">笑呵呵地</option>
  <!--此处省略了添加其他列表项的代码-->
  <option value="无精打采的">无精打采的</option>
</select>
说：  悄悄话
<input name="isPrivate" type="checkbox" class="noborder" id="isPrivate" value="true" onClick="checkIsPrivate()">
滚屏
<input name="scrollScreen" type="checkbox" class="noborder" id="scrollScreen" onClick="checkScrollScreen()" value= "1" checked>
    字体颜色：
      <select name="color" size="1" class="wenbenkuang" id="select">
        <option selected>默认颜色</option>
        <option style="color:#FF0000" value="FF0000">红色热情</option>
        <option style="color:#0000FF" value="0000FF">蓝色开朗</option>
        <!--此处省略了添加其他列表项的代码-->
        <option style="color:#999999" value="999999">烟雨蒙蒙</option>
      </select>
    <input name="content1" type="text" size="70" onKeyDown="if(event.keyCode==13 && event.ctrlKey) {send();}">
    <input name="Submit2" type="button" class="btn_blank" value="发送" onClick="send()">
    <input name="button_exit" type="button" class="btn_orange" value="退出聊天室" onClick="Exit()">
</form>
```

在上面的代码中，语句<%=session.getAttribute("username")%>用于显示当前的登录用户名。

当聊天对象文本框被设置为只读属性时，用户将不能手动输入聊天对象，因此还需要提供选择聊天对象的

功能。这可以通过在聊天室的主界面中添加选择聊天对象的 JavaScript 自定义函数及在在线人员列表中添加超链接来实现。实现将选择的聊天对象添加到聊天对象文本框的 JavaScript 代码如下：

```
function set(selectPerson){                    //自动添加聊天对象
    if(selectPerson!="<%=session.getAttribute("username")%>"){
        if(form1.isPrivate.checked && selectPerson=="所有人"){
            alert("请选择私聊对象！");
        }else{
            form1.to.value=selectPerson;
        }
    }else{
        alert("请重新选择聊天对象！");
    }
}
```

关于在线人员列表中添加超链接的代码可以参见实例 573。

（2）在聊天室的主界面中编写自定义的 JavaScript 函数 send()，用于调用 Ajax 实现用户发言。在该函数中，首先验证用户输入信息的合法性，然后再将提交的表单元素的内容连接为一个参数字符串，最后实例化 Ajax 对象。send()函数的具体代码如下：

```
function send(){            //验证聊天信息并发送
    if(form1.to.value==""){
        alert("请选择聊天对象！");return false;
    }
    if(form1.content1.value==""){
        alert("发送信息不可以为空！");form1.content1.focus();return false;
    }
    if(form1.isPrivate.checked){
        isPrivate="true";
    }else{
        isPrivate="false";
    }
    var param="from="+form1.from.value+"&face="+form1.face.value+"&color="+form1.color.value+
            "&to="+form1.to.value+"&content="+form1.content1.value+"&isPrivate="+isPrivate;
    var loader=new net.AjaxRequest("MessagesAction?action=sendMessage",deal_send,onerror,"POST",param);
}
```

（3）在聊天室相关的 Servlet 实现类中添加发送聊天信息的方法 sendMessages()。在该方法中，首先获取用户发言的相关信息，并对出现中文的信息进行转码；然后判断保存当前聊天信息的 XML 文件是否存在，如果不存在则创建该文件；最后将聊天信息保存到 XML 文件中，并重定向网页。sendMessages()方法的具体代码如下：

```
public void sendMessages(HttpServletRequest request,HttpServletResponse response){
    response.setContentType("text/html;charset=GBK");
    StringUtils su = new StringUtils();
    Random random = new Random();
    String from = su.toUTF8(request.getParameter("from"));                    //发言人
    String face = su.toUTF8(request.getParameter("face"));                    //表情
    String to = su.toUTF8(request.getParameter("to"));                        //接收者
    String color = request.getParameter("color");                             //字体颜色
    String content = su.toUTF8(request.getParameter("content"));              //发言内容
    String isPrivate = request.getParameter("isPrivate");                     //是否为悄悄话
    String sendTime = new Date().toLocaleString();                            //发言时间
    /********************开始添加聊天信息**********************************/
    String fileURL = createFile(request, response);                           //当文件不存在时创建该文件
    SAXBuilder builder = new SAXBuilder();
    Document feedDoc;
    try {
        feedDoc = builder.build(new File(fileURL));
        Element root = feedDoc.getRootElement();
        Element channel = root.getChild("messages");
        Element newNode = new Element("message");
        channel.addContent(newNode);                                          //创建 messages 节点
        Element fromNode = new Element("from").setText(from);
        newNode.addContent(fromNode);                                         //添加发言人子节点
        Element faceNode = new Element("face").setText(face);
        newNode.addContent(faceNode);                                         //添加表情子节点
        Element toNode = new Element("to").setText(to);
```

```
            newNode.addContent(toNode);                                    //添加接收者子节点
            Element contentNode = new Element("content").setText("<font color='"
                    + color + "'>" + content + "</font>");
            newNode.addContent(contentNode);                               //添加聊天内容子节点
            //发言时间
            Element sendTimeNode = new Element("sendTime").setText(sendTime);
            newNode.addContent(sendTimeNode);                              //添加发言时间子节点
            Element isPrivateNode = new Element("isPrivate").setText(isPrivate);
            newNode.addContent(isPrivateNode);                             //添加是否为悄悄话子节点
            request.getRequestDispatcher("MessagesAction?action=getMessages&nocache="
                            + random.nextInt(10000)).forward(request, response);
            XMLOutputter xml = new XMLOutputter(Format.getPrettyFormat());
            xml.output(feedDoc, new FileOutputStream(fileURL));
        /** ****************添加结束****************************** */
        } catch (Exception e) {
            e.printStackTrace();
        }
}
```

在上面的代码中调用了 createFile()方法，该方法用于根据当前日期生成 XML 文件名并判断该文件是否存在，如果不存在则创建该文件。具体代码如下：

```
public String createFile(HttpServletRequest request,HttpServletResponse response) {
        Date date = new Date();
        String newTime = new SimpleDateFormat("yyyyMMdd").format(date);
        String fileURL = request.getRealPath("xml/" + newTime + ".xml");          //生成文件名
        /****************判断 XML 文件是否存在，如果不存在则创建该文件***********/
        File file = new File(fileURL);
        if (!file.exists()) {                                          //判断文件是否存在，如果不存在则创建该文件
            try {
                file.createNewFile();                                  //创建文件
                String dataStr = "<?xml version=\"1.0\" encoding=\"GBK\"?>\r\n";
                dataStr = dataStr + "<chat>\r\n";
                dataStr = dataStr + "<messages></messages>";
                dataStr = dataStr + "</chat>\r\n";
                byte[] content = dataStr.getBytes();
                FileOutputStream fout = new FileOutputStream(file);
                fout.write(content);                                   //将数据写入输出流
                fout.flush();                                          //刷新缓冲区
                fout.close();                                          //关闭输出流
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return fileURL;
}
```

秘笈心法

心法领悟 574：通过快捷键发送聊天信息。

使用过 QQ 聊天软件的用户都知道，在通过 QQ 聊天时，编写好聊天内容后按 Ctrl+Enter 快捷键即可发送聊天内容。实际上，实现该功能很简单，只需在聊天内容文本框的 onKeyDown 事件中判断当前按下的快捷键是否为 Ctrl+Enter，如果是则调用自定义的 send()方法发送聊天内容。关键代码如下：

```
<input name="content1" type="text" size="70" onKeyDown="if(event.keyCode==13 && event.ctrlKey){send();}">alist.add(new Integer(1));
```

实例 575 实时显示聊天内容

光盘位置：光盘\MR\23\575

高级

实用指数：★★★★☆

在聊天室程序中，另一个必不可少的功能就是实时显示聊天内容。在聊天室的主界面中，用于显示聊天内容的部分大约占整个页面的 2/3，可见显示聊天内容的重要性。本实例将实现带有滚屏控制的显示聊天内容的功

能。进入聊天室的主界面，用户可以通过是否选中“滚屏”复选框来控制是否滚屏显示聊天内容。实例运行结果如图 23.3 所示。



图 23.3　实时显示聊天内容

■ 关键技术

本实例中应用的主要技术是 Ajax 重构和通过 JDOM 解析 XML 文件。关于 Ajax 重构的方法，参见实例 573；关于通过 JDOM 解析 XML 文件的具体介绍参见实例 574。

■

（1）创建一个单独的 JavaScript 文件，名称为 AjaxRequest.js，具体代码参见实例 573 的“关键技术”。

（2）在聊天室的主界面 main.jsp 中，应用代码中包含 AjaxRequest.js 文件。

（3）在聊天室的主界面中编写自定义的 JavaScript 函数 showContent()，用于实例化 Ajax 对象。showContent() 函数的具体代码如下：

```
function showContent(){
    var loader1=new net.AjaxRequest("MessagesAction?action=getMessages&nocache="+new Date().getTime(), deal_ content,onerror,"GET");
}
```

（4）从上面的代码中可以看出，Ajax 进行异步请求目标的 URL 地址为 MessagesAction?action=getMessages。从该 URL 地址可以看出，在进入主界面前会调用聊天室相关的 Servlet 实现类中的 getMessages()方法。在该方法中，首先判断保存当前聊天信息的 XML 文件是否存在，如果不存在则创建；然后解析保存聊天信息的 XML 文件，并将聊天内容连接为一个字符串；最后将连接后的字符串保存到 HttpServletRequest 对象中，并重定向网页到输入聊天内容的页面 content.jsp。getMessages()方法的具体代码如下：

```
public void getMessages(HttpServletRequest request,HttpServletResponse response) {
    response.setContentType("text/html;charset=GBK");
     String fileURL = createFile(request, response); //当文件不存在时创建该文件
    /******************开始解析保存聊天内容的 XML 文件*********************/
    try {
        SAXBuilder builder = new SAXBuilder();
        Document feedDoc = builder.build(new File(fileURL));
        Element root = feedDoc.getRootElement();                    //获取根节点
        Element channel = root.getChild("messages");                //获取 messages 节点
        Iterator items = channel.getChildren("message").iterator(); //获取 message 节点
        String messages = "";
        HttpSession session = request.getSession();                 //获取当前用户
        String userName = "";
        if (null == session.getAttribute("username")) {
            request.setAttribute("messages", "error");              //保存标记信息，表示用户账户已经过期
        } else {
            userName = session.getAttribute("username").toString();
            DateFormat df = DateFormat.getDateTimeInstance();
            while (items.hasNext()) {
                Element item = (Element) items.next();
                String sendTime = item.getChildText("sendTime");    //获取发言时间
                try {
                    if (df.parse(sendTime).after(
                            df.parse(session.getAttribute("loginTime").toString()))
                            || sendTime.equals(session.getAttribute("loginTime").toString())) {
                        String from = item.getChildText("from");       //获取发言人
                        String face = item.getChildText("face");       //获取表情
                        String to = item.getChildText("to");           //获取接收者
                        String content = item.getChildText("content"); //获取发言内容
```

```
                        boolean isPrivate = Boolean.valueOf(item.getChildText("isPrivate"));
                        if (isPrivate) {                                        //获取私聊内容
                            if (userName.equals(to) || userName.equals(from)) {
                                messages += "<font color='red'>[私人对话]</font>"+
                                "<font color='blue'><b>"+ from+
                                "</b></font><font color='#CC0000'>"+ face
                                        + "</font>对<font color='green'>["
                                        + to+ "]</font>说："+ content
                                        + " <font color='gray'>["
                                        + sendTime + "]</font><br>";
                            }
                        } else if ("[系统公告]".equals(from)) {                 //获取系统公告信息
                            messages += "[系统公告]：" + content
                                    + " <font color='gray'>["
                                    + sendTime + "]</font><br>";
                        } else {                                                //获取普通发言信息
                            messages += "<font color='blue'><b>" + from
                                    + "</b></font><font color='#CC0000'>"
                                    + face+ "</font>对<font color='green'>[" + to
                                    + "]</font>说：" + content
                                    + " <font color='gray'>["
                                    + sendTime + "]</font><br>";
                        }
                    }
                } catch (Exception e) {
                    System.out.println("" + e.getMessage());
                }
            }
            request.setAttribute("messages", messages);                         //保存获取的聊天信息
        }
        request.getRequestDispatcher("content.jsp").forward(request,response);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

（5）编写显示聊天内容的 JSP 页面 showEmailNumber.jsp，在该页面中只需要应用 out 对象的 println()方法将返回的执行结果输出即可。具体代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java" import="java.util.*" errorPage="" %>
<% request.setCharacterEncoding("gbk"); %>
<%out.println(request.getAttribute("messages").toString());
%>
```

（6）在聊天室的主界面中，在右侧显示聊天内容的单元格中添加一个 id 属性为 content 的<div>标签，用于实时显示聊天内容。具体代码如下：

```
<div style="height:290px; overflow:hidden" id="content">聊天内容</div>
```

（7）在聊天室的主界面中，编写 Ajax 的回调函数 deal_content()。在该函数中，首先获取 Ajax 处理页的返回值；然后去除字符串中的 Unicode 空白符；最后判断在获取信息时是否产生错误，如果是则退出聊天室，否则将获取的聊天内容赋值给 id 为 content 的<div>标签的 innerHTML 属性。deal_content()函数的具体代码如下：

```
function deal_content(){
    var returnValue=this.req.responseText;                  //获取 Ajax 处理页的返回值
    var h=returnValue.replace(/\s/g,"");                    //去除字符串中的 Unicode 空白符
    if(h=="error"){                                         //判断在获取信息时是否产生错误，如果是则退出聊天室
        Exit();
    }else{
        content.innerHTML=sysBBS+returnValue+"</span>";     //显示聊天内容
    }
}
```

（8）为了让页面载入后就调用 Ajax 获取聊天内容，并且每隔一秒钟便获取一次数据，还需要在聊天室的主界面中添加以下 JavaScript 代码：

```
window.setInterval("showContent();",1000);
window.onload=function(){
    showContent();                                          //当页面载入后显示聊天内容
}
```

至此，聊天内容即可实时地显示在聊天室的主界面中。接下来，还需要实现滚屏控制。

（9）在聊天室的主界面中添加一个由用户控制是否滚屏的复选框，在该复选框的 onClick 事件中调用一个用于控制是否滚屏的方法 checkScrollScreen()。具体代码如下：

```
<input name="scrollScreen" type="checkbox" class="noborder" id="scrollScreen" onClick="checkScrollScreen()" value= "1" checked>
```

（10）编写一个自定义的 JavaScript 函数 checkScrollScreen()，用于控制是否滚屏。在该方法中，首先判断步骤（9）添加的复选框是否为选中状态，如果不是则表示不滚屏，因此为显示聊天内容的<div>标签设置滚动条，否则将聊天内容的<div>标签的 scrollTop 属性设置为其滚动高度乘以 2。checkScrollScreen()函数的具体代码如下：

```
<script language="javascript">
function checkScrollScreen(){
    if(!form1.scrollScreen.checked){
        document.getElementById("content").style.overflow='scroll';
    }else{
        document.getElementById("content").style.overflow='hidden';
        //当聊天信息超过一屏时，设置最先发送的聊天信息不显示
        document.getElementById('content').scrollTop = document.getElementById('content').scrollHeight*2;
    }
    setTimeout('checkScrollScreen()',500);
}
</script>
```

秘笈心法

心法领悟 575：解决聊天内容中出现的少数汉字乱码的情况。

在开发聊天室时笔者采用的是 GB2312 编码，开始测试时，录入的一般都是比较常见的信息（例如“您好”），程序运行正常，但是当输入“槑”时出现了乱码。

要解决该问题，首先需要了解一下 GB2312 编码。GB2312 又称国标码，由国家标准化管理委员会发布。在国标码的字符集中共收录了一级汉字 3755 个，二级汉字 3008 个，图形符号 682 个，3 项字符总计 7445 个。该字符集中只包括常用的汉字，对于一些生僻字（例如“槑”），该字符集并不支持。这就是当输入“槑”字时会出现乱码的原因。要解决该问题，可以将程序的编码格式统一改为 GBK。GBK 编码是汉字国标扩展码，基本上采用了 GB2312 中所有的汉字及码位，并涵盖了 Unicode 中所有的汉字，总共收录了 3883 个符号、21003 个汉字及 1894 个造字位。GBK 是简体字和繁体字融于一库的编码格式。

实例 576	安全退出聊天室 光盘位置：光盘\MR\23\576	高级 实用指数：★★★★☆

在聊天室中，通常会有两种退出聊天室的方法，一种是单击主界面中的“退出聊天室”按钮，另一种是单击浏览器的“关闭”按钮☒。本实例也充分考虑了这两点，提供了两种退出聊天室的途径。但是无论采用哪种方法，都会弹出如图 23.4 所示的对话框。



图 23.4　安全退出聊天室

关键技术

在实现本实例时，主要用到了 Session 销毁技术和 JavaScript 技术。下面对 Session 销毁技术进行介绍。

当用户退出聊天室后，需要将保存用户登录信息的 Session 销毁，这样才能保证聊天室的实效性。通过 Session 对象的 invalidate()方法可以销毁 Session。语法如下：

```
session.invalidate();
```

Session 被销毁后，就不能再使用该 Session 对象了。如果在 Session 被销毁后，再次调用 Session 对象的任何方法，都将报出 Session already invalidated 异常。

设计过程

（1）在聊天室的主界面的合适位置添加“退出聊天室”按钮，并在按钮的 onclick 事件中调用自定义的 JavaScript 函数 Exit()。关键代码如下：

```
<input name="button_exit" type="button" class="btn_orange" value="退出聊天室" onClick="Exit()">
```

（2）在聊天室的主界面中，编写自定义的 JavaScript 函数 Exit()。在该函数中首先将页面重定向到退出聊天室页面 leave.jsp，然后弹出“欢迎您下次光临！”对话框。具体代码如下：

```
function Exit(){
    window.location.href="leave.jsp";
    alert("欢迎您下次光临！");
}
```

（3）编写退出聊天室页面 leave.jsp。在该页面中，首先销毁 Session，然后将页面重定向到登录页面。leave.jsp 页面完整代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java" import="java.util.*"%>
<% request.setCharacterEncoding("gbk");
session.invalidate();                              //销毁 Session
response.sendRedirect("index.jsp");                //重定向页面到登录页面
%>
```

（4）实现单击浏览器中的“关闭”按钮时退出聊天室，代码如下：

```
<script language="jscript">
window.onbeforeunload=function(){                  //当用户单击浏览器中的“关闭”按钮时，执行退出操作
        ver = navigator.appVersion;                //浏览器的版本
        bType =navigator.appName;                  //浏览器的类型
        vNumber=parseFloat(ver.substring(ver.indexOf("MSIE")+5,ver.lastIndexOf("Windows")));
        if(bType=="Microsoft Internet Explorer"){
                if(vNumber>6){                     //IE6 以上浏览器
                        vOffset=document.body.offsetWidth-document.body.scrollWidth;
                        if(event.clientY<0 && event.clientX>document.body.scrollWidth-vOffset-20){
                                Exit();            //执行退出操作
                        }else{                     //IE6 及以下浏览器
                                if(event.clientY<0 && event.clientX>document.body.scrollWidth){
                                        Exit();    //执行退出操作
                                }
                        }
                }
        }
}
```


心法领悟 576：判断 Ajax 返回结果是否等于指定字符串。

在应用 Ajax 技术时，有时需要判断返回结果是否等于某个字符串。例如，判断返回结果是否等于字符串“很抱歉，发言失败！”，可以使用以下代码：

```
var h=this.req.responseText;
h=h.replace(/\s/g,"");                  //去除字符串中的 Unicode 空白符
if(h!="很抱歉，发言失败！"){
        ……
}
```

这是因为在返回结果中包括 Unicode 空白符，如果不去除这些空白符，在判断时将无法得到对应的值。

23.2　博客网核心模块开发

实例 577　**注册自己的博客**　　高级

光盘位置：光盘\MR\23\577　　实用指数：★★★★

实例说明

经常上网的读者一般都会拥有自己的博客，那么对于注册博客也一定不陌生。一些大型的门户网站或专业的博客网站都为用户提供了注册其博客的功能。本实例将介绍如何在博客网中注册自己的博客。运行程序，单击“注册”按钮，将打开注册页面，正确填写用户信息（如图 23.5 所示）后，单击“添加”按钮，系统将对输入信息进行校验，如果输入的信息合法，将完成用户注册，否则给出提示。

图 23.5　注册自己的博客

关键技术

本实例主要应用 Struts2 及其提供的标签实现用户注册和表单校验。关于 Struts2 的介绍可参见第 15 章。

（1）编写用户注册页面 addUserInfo.jsp，在其中应用 Struts2 标签添加收集用户注册信息的表单。关键代码如下：

```
<%@ taglib prefix="s" uri="/struts-tags"%>
<s:form action="userInfo_addUserInfo">
 <tr >
  <td width="73" height="30" bgcolor="F9F9F9">用户名</td>
  <td width="288" height="30">
      <s:textfield name="account"/><s:fielderror>
      <s:param value="%{'account'}"/></s:fielderror></td>                    //用户名标签的设置
  <td width="82" height="30">真实姓名</td>
  <td width="303" height="30">
      <s:textfield name ="realname"/><s:fielderror>
      <s:param value="%{'realname'}"/></s:fielderror></td>                   //用户真实姓名标签的设置
 </tr>
      ……//省略其他表单的设置
 <tr bgcolor="#FFFFFF" >
  <td height="30" colspan="4" align="center"><s:submit value="  添加  "/>
      <s:hidden name="homepage" value="%{# request.homepage}"/></td>      //提交按钮标签的设置
 </tr>
</s:form>
<s:fielderror><s:param value="%{'reepassword'}"/></s:fielderror>
```

（2）对用户填写的注册信息进行校验。为了实现校验执行指定处理逻辑的功能，Struts2 的 Action 类允许提供一个 validatexxx()方法，其中 xxx 即 Action 类对应的处理逻辑方法。在 addUserInfo()方法前，执行用户注册信息校验的方法 validateAddUserInfo()的关键代码如下：

```
public void validateAddUserInfo() {
      objectDao = new ObjectDao<UserInfo>();                                        //实例化 ObjectDao
```

```
        if (this.userInfo.getAccount().equals("")) {                              //验证用户名是否为空
                this.addFieldError("account", "用户名不能为空！");
        } else {
                objectDao = new ObjectDao<UserInfo>();
                hql = "from UserInfo where account= '" + userInfo.getAccount()+ "'";    //查询用户名是否存在的 hql
                if (null != objectDao.queryFrom(hql)) {                           //用户名唯一性验证
                this.addFieldError("account", "用户名重复，请重新输入！");
                }
        }
        if (this.userInfo.getRealname().equals("")) {                             //验证真实姓名是否为空
                this.addFieldError("realname", "用户真实姓名不能为空！");
        }
        ……//省略其他字段验证
}
```

在上述代码中，一旦判断表单信息不符合要求，就会把校验失败的提示信息通过 addFieldError()方法添加到 fieldError 信息中，之后系统会自动返回 input 逻辑视图，该逻辑视图需要在 struts.xml 配置文件中进行配置。

为了在 input 视图对应的 JSP 页面中输出错误提示，应该在页面中编写如下标签代码：

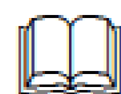
```
<!-- fielderror 标签专门负责输出系统的 fieldError 信息，也就是输出校验失败提示 -->
<s:fielderror/>
```

（3）如果校验用户注册表单成功，则直接进入业务逻辑处理的 addUserInfo()方法。该方法的主要代码如下：

```
public String addUserInfo() {
        objectDao = new ObjectDao<UserInfo>();
        userInfo.setPassword(com.mr.tools.ValidateExpression.encodeMD5(userInfo.getPassword()));
        userInfo.setHomepage(userInfo.getHomepage() + userInfo.getAccount());
        boolean flag = objectDao.saveT(userInfo);
        String result = "";
        if (flag) {
                //将模板中的 index.jsp 页面保存在用户名文件夹下
                String descPath = ServletActionContext.getRequest().getRealPath("/" + userInfo.getAccount() + "");
                String sourPath = ServletActionContext.getRequest().getRealPath("/templet/index.jsp");
                if (com.mr.tools.FileOperation.buildJSP(sourPath, descPath,userInfo.getAccount())) {
                        result = "您注册成功！";
                        request.getSession().setAttribute("freeze",userInfo.getFreeze());
                        request.getSession().setAttribute("account",userInfo.getAccount());
                }
        }
        request.setAttribute("result", result);
        request.setAttribute("sign", "1");
        return "operationUser";
}
```

（4）将模板文件保存在当前用户文件夹下。当用户注册成功后，根据用户名在服务器端创建指定的文件夹，并且将模板文件 index.jsp 复制到该文件夹下。其中创建与复制文件应用 buildJSP()方法实现，其关键代码如下：

```
fileinputstream = new FileInputStream(souPath);                    //获取模板文件的路径
bytes = new byte[1024 * 5];
fileinputstream.read(bytes);
fileinputstream.close();
String templateContent = new String(bytes);
File file = new File(desPath);
if(!file.exists()){                                                 //创建文件夹
      file.mkdir();
}
desPath=desPath+"/index.jsp";                                       //将模板创建或复制到指定文件夹下
FileOutputStream fileoutputstream = new FileOutputStream(desPath);
byte tag_bytes[] = templateContent.getBytes();
fileoutputstream.write(tag_bytes);
fileoutputstream.close();
```

**说明：** 模板文件保存在服务器端的 templeate 文件夹中，其名称为 index.jsp，主要作用是获取域名中的用户名，进入该用户的博客空间内。

index.jsp 模板文件的代码如下：

```
<%
      String path1 = request.getServletPath();
            path1 = path1.substring(1);
```

```
        path1 = path1.substring(0, path1.indexOf("/"));
        response.sendRedirect("userInfo_goinUser.htm?account=" + path1 + "");
%>
```

说明： 用户注册完成后，实际上就已经拥有了自己的博客，但是这时还不能访问，因为刚注册的用户博客是被系统冻结的，需要管理员解冻后才能访问。

秘笈心法

心法领悟 577：解决生成验证码时缓存图片的问题。

在生成彩色图文验证码时，实现的单击“看不清？换一个”超链接时显示新的验证码功能，在 IE 浏览器中运行一切正常，但是在火狐等浏览器中，将出现验证码不改变的情况。这是因为在火狐浏览器中，缓存了验证码图片。这时可以在请求处理页面中，将图片的 src 属性通过 JSP 代码动态传递一个参数，参数值为随机数即可。例如下面的代码：

```
<%@ page contentType="text/html; charset=GBK" language="java"    import="java.util.Random"%>
<%
Random random = new Random();
%>
<img src="<% out.println("PictureCheckCode?rand="+random.nextInt(10000));%>" id="createCheckCode" width="200" height="60">
<a href="#" style="color:#EEEEEE" onclick="getCheckCode1(showCheckCode,checkCode)">看不清?换一个</a>
```

实例 578	根据域名访问博客 光盘位置：光盘\MR\23\578	高级 实用指数：★★★★

实例说明

在实例 577 中介绍了如何注册自己的博客，本实例将介绍如何根据域名访问博客。根据域名访问博客实际上就是根据浏览器 IE 地址访问相应的用户博客。如图 23.6 所示，浏览器的 IE 地址是 http://192.168.1.66:8080/578/mrwgh/index.jsp，其中 mrwgh 为博客的用户名，根据这个用户名访问用户博客，运行结果如图 23.7 所示。

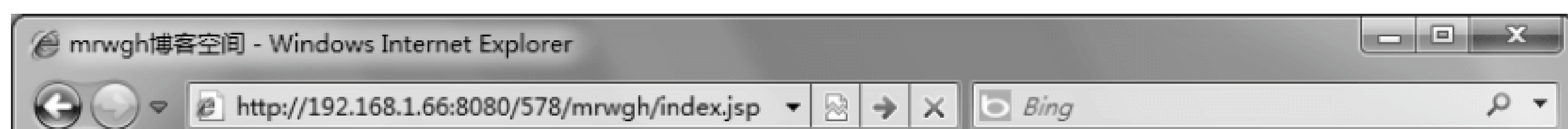


图 23.6 在 IE 浏览器中输入博客地址



图 23.7 访问用户 mrwgh 的博客

本实例主要应用了 Struts2 和 Hibernate 整合技术。关于 Struts2 的介绍参见第 15 章；关于 Hibernate 的介绍参见第 16 章。

设计过程

从实例 577“注册自己的博客”的实现过程中可以知道，当用户注册成功后，服务器端将根据用户名创建指定的文件夹并且将模板文件 index.jsp 复制到该文件夹下，当用户在 URL 地址中加入用户名，则系统将自动访问用户名文件夹下的 index.jsp。

index.jsp 页面中的重定向请求地址是 userInfo_goinUser.htm?account="+path1+"，根据 struts.xml 文件的配置信息可以知道，该请求地址执行的是 UserInfoAction 类中的 goinUser()方法。该方法的代码如下：

```
public String goinUser() {
    String account = request.getParameter("account");
    try {
        account = new String(account.getBytes("ISO8859_1"), "gb2312");    //对用户名转码
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
    hql = "from UserInfo where account = '" + account + "'";            //根据用户名查询用户信息的 HQL 语句
    objectDao = new ObjectDao<UserInfo>();
    UserInfo userInfo = objectDao.queryFrom(hql);
    if (userInfo.getFreeze().equals("冻结")) {                           //判断当前用户是否被冻结
        request.setAttribute("sign", "7");
        return "operationUser";
    } else {
        objectDao = new ObjectDao<UserInfo>();
        if (!userInfo.getAccount().equals(account)) {
            userInfo.setVistor(userInfo.getVistor() + 1);
            objectDao.updateT(userInfo);
        }
        request.getSession().setAttribute("userInfo", userInfo);
    }
    return "goinUser";
}
```

goinUser()方法的代码执行流程如图 23.8 所示。

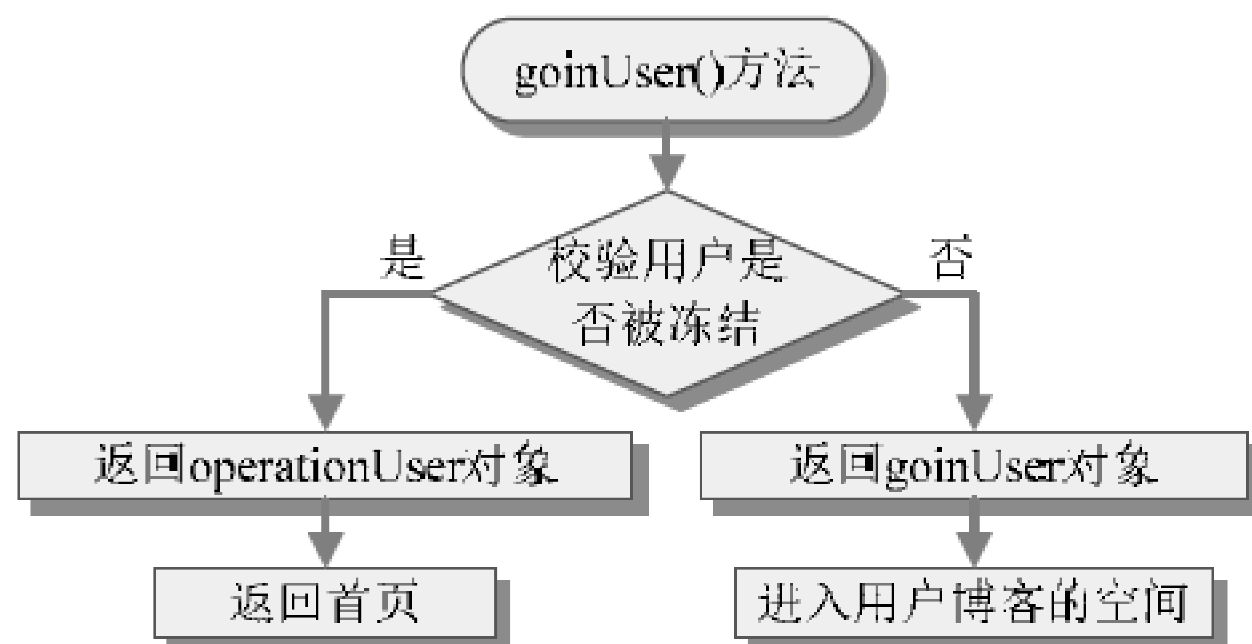


图 23.8　goinUser()方法的代码执行流程

秘笈心法

心法领悟 578：Struts2 中的 JAR 包。

大部分情况下，使用 Struts2 的 Web 应用并不需要用到 Struts 的全部特性，因此没有必要一次将该 lib 路径下的 JAR 文件全部复制到 Web 应用的 WEB_INF\lib 路径下。

实例 579　推荐博客设置

光盘位置：光盘\MR\23\579

高级

实用指数：★★★

在博客网中，为了提升某个博客的人气，使其被更多用户知道，常常将该博客列为网站所推荐的博客，在首页中显示其名称，如图 23.9 所示。推荐博客设置属于网站的后台操作，当管理员成功登录后台后，单击导航

区域中的“用户管理”超链接，将显示用户信息列表，如图 23.10 所示。在该列表中，有一列“是否推荐”，单击某个用户信息后面的超链接，即可将该博客设置为被推荐博客。



图 23.9　推荐博客

用户查询

	编号	用户名	Email地址	真实姓名	性别	博客点击率	是否推荐	冻结/解冻
□	7	mrwgh	wgh717@sohu.com	wanggh	女	0	是	解冻
□	6	sss	ssss@163.com	dsa	男	0	否	冻结

图 23.10　后台显示用户信息列表页面

关键技术

推荐博客设置主要是根据用户信息表（tb_userInfo）中的 commend 字段值的不同进行判断，当该字段值为“是”，则说明当前用户的博客空间属于推荐状态；当该字段值为“否”，则说明当前用户的博客空间属于未被推荐状态。实现推荐博客主要应用的是 Hibernate 技术对数据表进行操作，关于 Hibernate 技术的具体介绍参见第 16 章。

设计过程

设置博客推荐状态可通过 UserInfoAction 类中的 updateCommendUser()方法实现，该方法主要用于更改当前用户中的 commend 字段的值。updateCommendUser()方法的主要代码如下：

```
public String updateCommendUser() {
        objectDao = new ObjectDao<UserInfo>();                              //实例化 ObjectDao
        Integer id = Integer.valueOf(request.getParameter("id"));           //获取 id 参数
        hql = "from UserInfo where id = " + id + "";                        //根据 id 值查找 User 对象
        UserInfo userInfo = objectDao.queryFrom(hql);
        if (userInfo.getFreeze().equals("解冻")) {                          //判断该用户是否被解冻
                if (userInfo.getCommend().equals("是")) {                   //判断该用户是否推荐
                        userInfo.setCommend("否");
                } else if (userInfo.getCommend().equals("否")) {
                        userInfo.setCommend("是");
                }
        } else {
                request.setAttribute("result", "该用户没有被解冻，不能够推荐！ ");
        }
        boolean flag = objectDao.updateT(userInfo);                         //修改用户博客的推荐状态
        if (flag) {
                Integer pageCounter = Integer.valueOf(request.getParameter("count"));
                request.setAttribute("pageCounter", pageCounter);
        }
        request.setAttribute("sign", "5");
        return "operationUser";
}
```

updateCommendUser()方法的代码执行流程如图 23.11 所示。

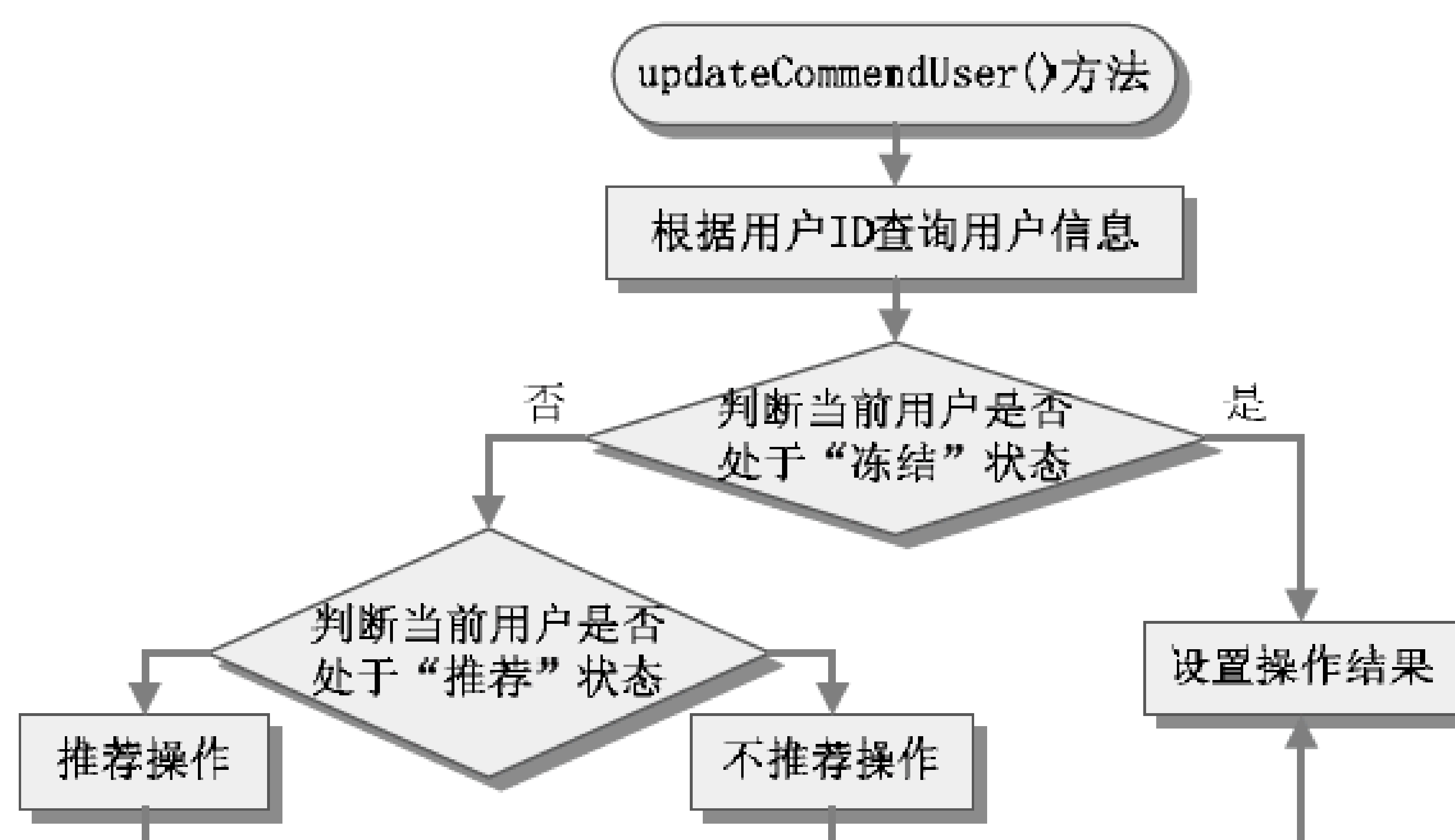


图 23.11　updateCommendUser()方法的代码执行流程

秘笈心法

心法领悟 579：Struts2 的 Action 配置。

Struts2 的 Action 默认拦截所有后缀为.action 的请求，因此如果程序员需要将某个表单提交给 Struts2 的 Action 处理，则应该将该表单的 action 属性设置为*.action 的格式。

实例 580　文章浏览操作

光盘位置：光盘\MR\23\580

高级

实用指数：★★★

实例说明

在博客网中，文章（也就是用户发表的博文）是必不可少的。本实例将介绍如何实现博客网中的文章浏览操作。该功能是博客网的后台操作，管理员成功登录网站后台后，单击导航区域中的“文章管理”超链接，即可进入文章浏览页面。运行本实例，将显示如图 23.12 所示的运行结果，单击左侧的用户名，可以查看对应用户的文章。

mrwgh

test

文章查询

文章标题	文章类型	文章发布人	文章时间	文章访问次数	操作
Spring技术入门	技术	test	2010-04-22 09:14:52	0	详细查询
Hibernate技术入门	笑话	test	2010-04-22 09:16:19	0	详细查询
struts2高级技术应用	技术	test	2010-04-22 09:16:19	0	详细查询

图 23.12　文章浏览

关键技术

本实例主要应用了 Struts2 和 Hibernate 整合技术。关于 Struts2 的介绍参见第 15 章；关于 Hibernate 的介绍参见第 16 章。

（1）利用 ArticleAction 类中的 admin_articleQuery()方法实现文章的查询。该方法主要实现 3 个功能：设置查询文章信息的 HQL 语句；执行 HQL 语句实现对文章的查询操作；对查询结果进行分页操作。代码如下：

```
public String admin_articleQuery() {
	//以下是对文章的全部查询
	hql = "from ArticleInfo";                              //设置对文章全部进行查询的 HQL 语句
	String account = request.getParameter("account");      //页面中的 account 参数
	if (null != account) {                                 //判断 account 参数是否为空
		hql = "from ArticleInfo where author = '" + account + "'";
		request.setAttribute("account", account);
	}
	objectDao = new ObjectDao<ArticleInfo>();              //持久化类 objectDao 对象的实例化
	List<ArticleInfo> list = objectDao.queryList(hql);     //执行查询的 HQL 语句
	//对分页进行操作
	int showNumber = 10;
	Integer count = 0;
	if (null != request.getParameter("count")) {
		count = Integer.valueOf(request.getParameter("count"));
	}
	list = objectDao.queryList(hql);
	int maxPage = list.size();
	if (maxPage % showNumber == 0) {
		maxPage = maxPage / showNumber;
	} else {
		maxPage = maxPage / showNumber + 1;
	}
	if (0 == count) {
		list = objectDao.queryList(hql, showNumber, count);
	} else {
```

```
                count--;
                list = objectDao.queryList(hql, showNumber, count * showNumber);
        }
        request.setAttribute("count", count);
        request.setAttribute("list", list);
        request.setAttribute("maxPage", maxPage);
        //文章所对应的发布人
        hql = "select author from ArticleInfo group by author";
        List authorList = objectDao.queryListObject(hql);
        request.setAttribute("authorList", authorList);
        return "admin_articleQuery";
}
```

admin_articleQuery()方法的代码执行流程如图 23.13 所示。

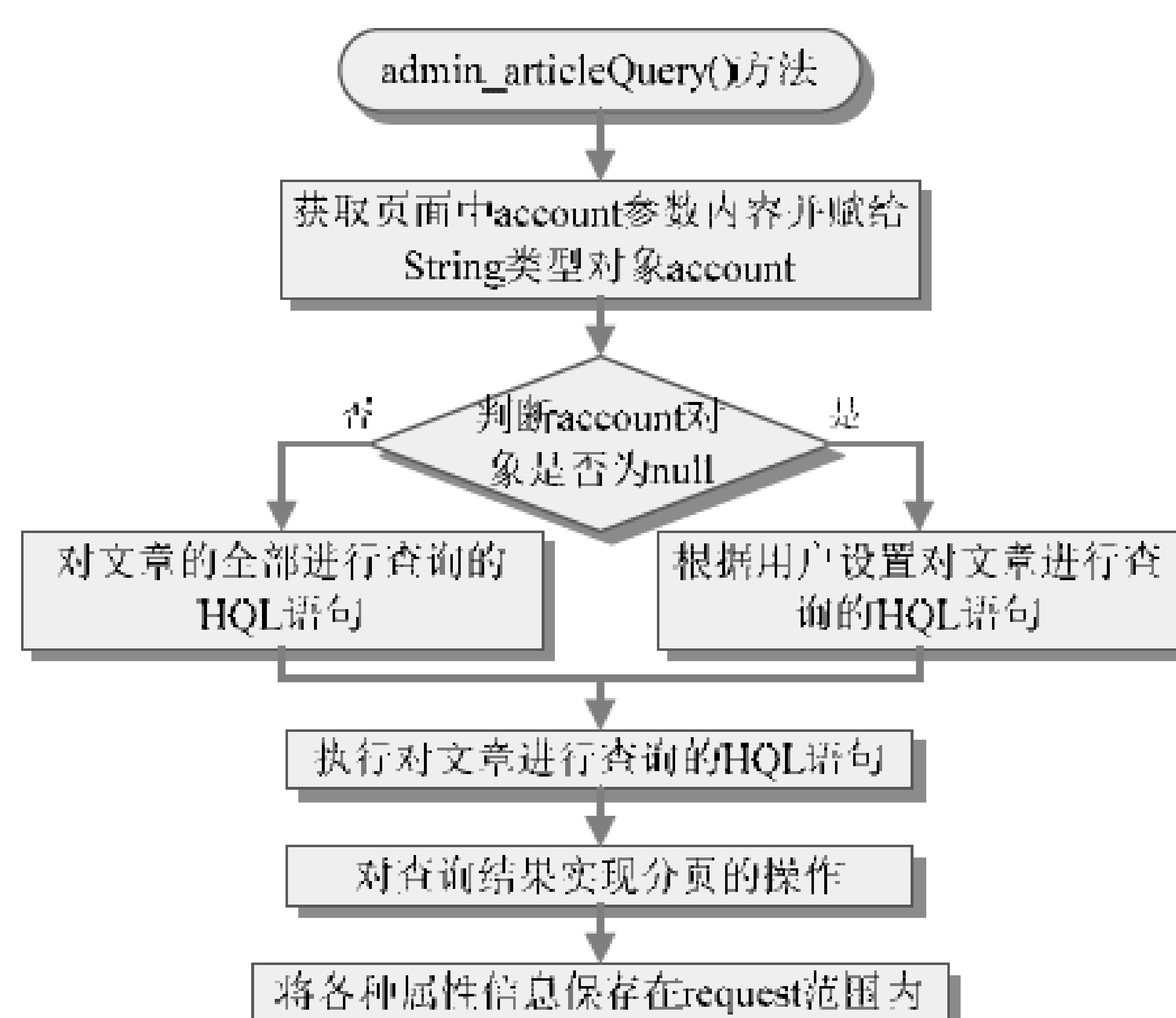


图 23.13　admin_articleQuery()方法的代码执行流程

（2）显示查询到的文章列表。从步骤（1）中可以知道 admin_articleQuery()方法的返回值类型是 String 类型，在 struts.xml 配置文件中根据该方法返回值内容转发到浏览文章页面，该页面主要显示查询到的文章的各种信息。关键代码如下：

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<!--显示用户的名称-->
        <s:iterator value="%{#request.authorList}" id="author">
        <tr>
          <td height="30" align="center" bgcolor="#FFFFFF"><s:a href="articleInfo_admin_article Query.htm?account =%{#author}"><s:property value="#author"/></s:a></td>
        </tr>
         </s:iterator>
<!--显示文章的主要内容-->
        <table width="692" border="0" cellspacing="0" cellpadding="0">
          <tr align="center">
            <td width="130" height="25">文章标题</td>
            <td width="134">文章类型</td>
            <td width="96">文章发布人</td>
            <td width="122">文章时间</td>
            <td width="93">文章访问次数</td>
            <td width="100">操作</td>
          </tr>
        <s:iterator value="%{#request.list}" id="article">
          <tr align="center">
            <td height="25"><s:property value="#article.title"/></td>
            <td><s:property value="#article.typeName"/></td>
            <td><s:property value="#article.author"/></td>
            <td><s:property value="#article.sendTime"/></td>
            <td><s:property value="#article.visit"/></td>
            <td><s:a href="articleInfo_admin_articleQueryOne.htm?id=%{#article.id}">详细查询</s:a></td>
          </tr>
          </s:iterator>
        </table>
<!--显示文章的分页操作-->
```

```
<s:bean name="org.apache.struts2.util.Counter" id="counter">
      <s:param name = "first" value="1"/>
      <s:param name = "last" value="%{#request.maxPage}"/>
      <s:iterator status="st" id="idd">
        <s:if test="%{#request.account==null}">
          <s:a href="articleInfo_admin_articleQuery.htm?count=%{idd}"><s:property /></s:a>
      </s:if>
      <s:else>
        <s:a href="articleInfo_admin_articleQuery.htm?count=%{idd}&account=%{#request.account}"><s:property /></s:a>
      </s:else>
      </s:iterator>
</s:bean>
```

秘笈心法

心法领悟 580：Action 的 name 属性的命名规则。

Action 的 name 命名是非常灵活的，但如果为 name 属性分配一个带点（.）或者带中划线（-）的值，例如，my.user 或者 my-action 等，则可能引发一些未知的异常。因此，不推荐带有点号和中划线的 name 属性。

23.3　在线投票统计功能

实例 581	实现投票功能 光盘位置：光盘\MR\23\581	中级 实用指数：★★★★☆

实例说明

数据的收集和分析是决策的基础，而通过网络收集数据是当前比较常用的一种形式。利用网络收集数据通常采用投票的形式，本实例将实现投票功能，运行结果如图 23.14 所示。

图 23.14　实现投票功能

关键技术

本实例在设计投票页面时，使用了 JSTL 标签进行控制，这样可以使页面更加灵活。

设计过程

（1）如图 23.14 所示的投票区由如下代码实现，所有参与投票的选项的信息保存在数据库中，这里通过循环将所有参与投票的选项显示到页面中。关键代码如下：

```
<form action="vote" name="voteform" method="post" target="resultpage">
<table border="0" width="100%">
      <tr height="95" align="center"><td colspan="2"><img src="images/lefttopbg.jpg"></td></tr>
      <c:set var="options" value="${requestScope.optionlist}"/>
      <c:if test="${empty options}">
      <tr><td colspan="2">没有投票选项</td></tr>
      </c:if>
      <c:if test="${!empty options}">
      <c:forEach var="option" varStatus="ovs" items="${options}">
      <tr>
            <td style="padding-left:20"><img src="images/title.jpg"> ${option.optionName}</td>
            <td align="center"><input type="radio" name="movie" value="${option.id}"
                  onclick="message.innerHTML=''"></td>
      </tr>
      <tr><td colspan="2"><img src="images/line.jpg"></td></tr>
      </c:forEach>
      </c:if>
```

```
        <tr height="40">
            <td><b><span id="message" style="color:red"></span></b></td>
            <td><input type="button" value="" name="voteb"
                style="background-image:url(images/submit.jpg);border:0;width:76;height:23"
                onclick="checkvote()"></td>
        </tr>
</table>
</form>
```

（2）单击“投票”按钮，将执行 VoteServlet 类中的 doPost()方法，在该方法中将调用 vote()方法提交投票。doPost()方法的完整代码如下：

```
protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {
        width = 0;
        height = 0;
        String servletPath = request.getServletPath();
        if ("/vote".equals(servletPath))                    //由提交投票触发
            vote(request, response);
        else if ("/showresult".equals(servletPath))         //由查看结果触发
            showresult(request, response);
}
```

秘笈心法

心法领悟 581：判断 IP 所属地区技术。

为了使投票结果更具权威性，可以通过 IP 地址判断出其所属的地区，例如，IP 地址 221.8.65.74 属于吉林省，这是因为为每个地区分配了一个或几个 IP 段，而吉林省的 IP 段之一为从 3708289023~3708420094。一个 IP 地址由 4 个段位组成，而上面给出的 IP 段是一个整数区间，下面将详细介绍将一个 IP 地址转换为其对应整数的方法。

在将 IP 地址转换为其对应的整数时，需要将 IP 地址的每个段位转换成二进制数，不足 8 位的在左边用 0 补齐，然后依次将这 4 个二进制数首尾连接，将得到一个完整的二进制流，将该二进制流转换为对应的十进制数，就是该 IP 地址所对应的整数。

实例 582	实现柱形图统计功能 光盘位置：光盘\MR\23\582	中级 实用指数：★★★★

当利用柱形图按投票项进行统计时，采用的是垂直绘制的柱形图，具体效果如图 23.15 所示；当利用柱形图按投票地区进行统计时，采用的是水平绘制的柱形图，具体效果如图 23.16 所示。

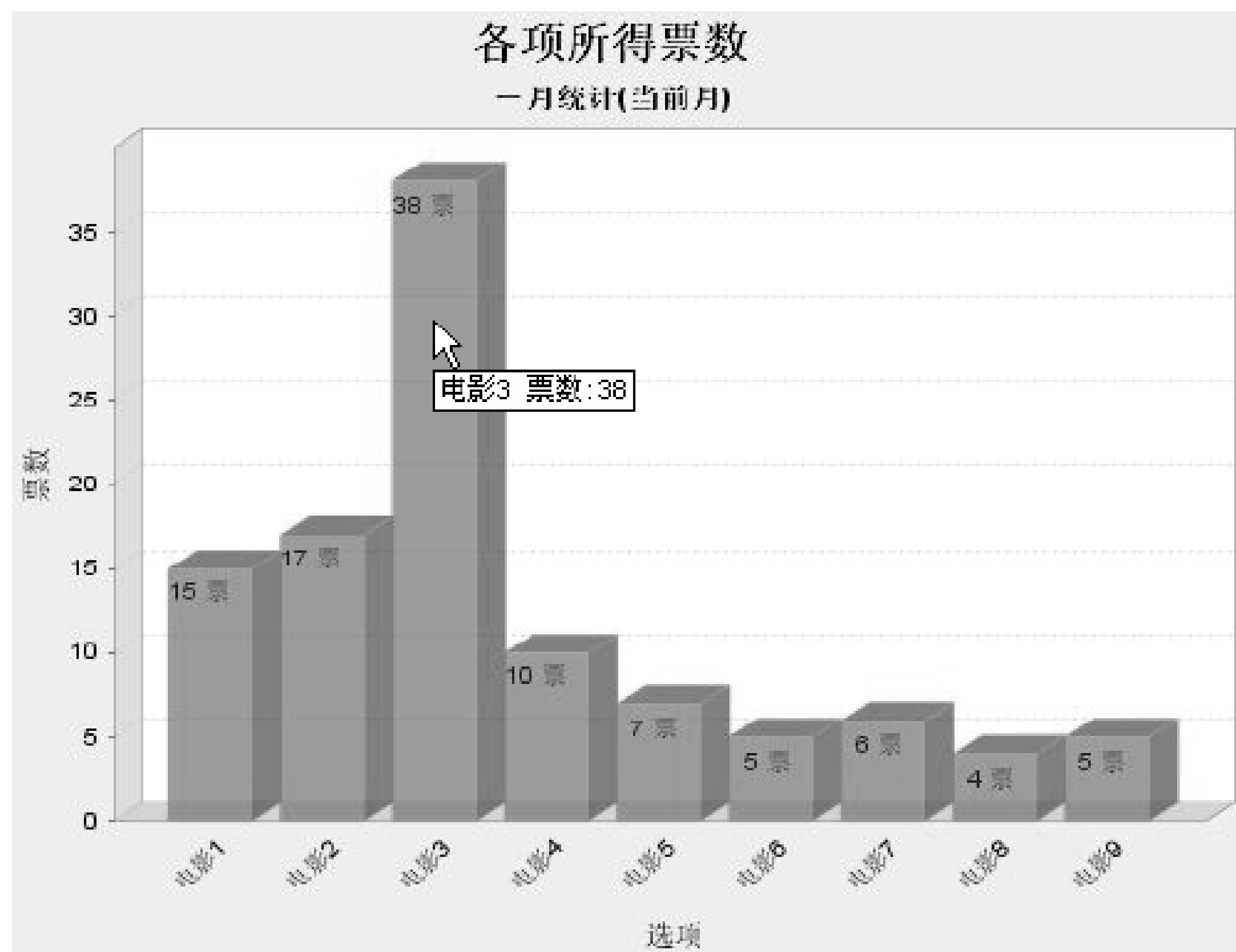


图 23.15　垂直绘制的柱形图

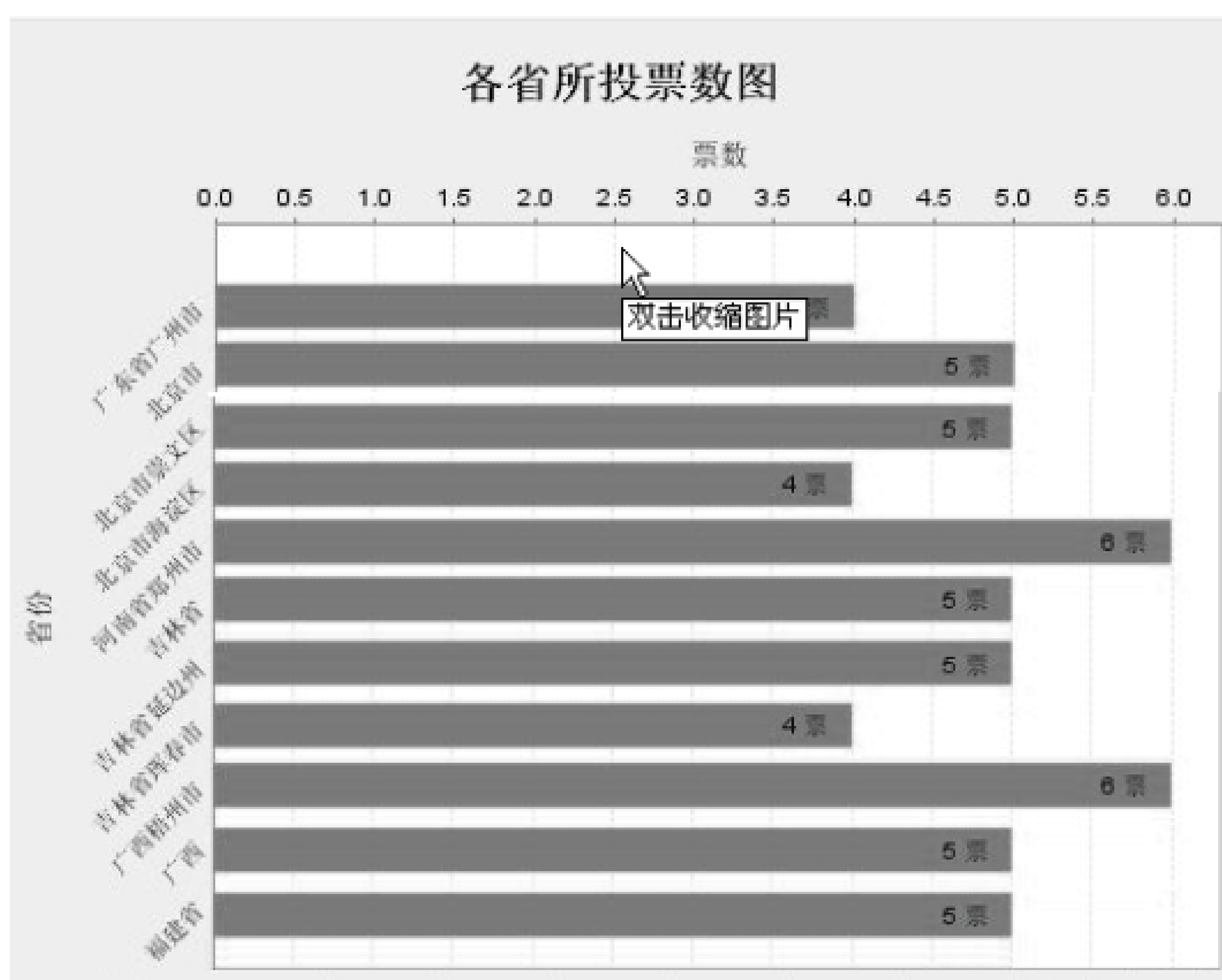


图 23.16　水平绘制的柱形图

关键技术

本实例使用了 JFreeChar 绘制统计图。本书在第 7 章曾重点讲解了如何利用 JFreeChar 绘制各种图表，读者可参考。

设计过程

（1）利用 VoteServlet 类中的 getChartForBar(String action, String method)方法创建柱形统计图对象的完整代码如下：

```
private JFreeChart getChartForBar(String action, String method) {
        CategoryDataset dataset = null;
        JFreeChart chart = null;
        String title1 = "";
        String title2 = "";
        String subtitle = "";
        if ("day".equals(method))
                subtitle = "一日统计(今日)";
        else if ("month".equals(method))
                subtitle = "一月统计(当前月)";
        if ("area".equals(action)) {                                    //处理查看“各省所投票数”的请求
                dataset = getDataSetForBarAndArea(method);              //获取数据集
                title1 = "各省所投票数图";
                title2 = "省份";
                width = 500;
                height = 100 + 25 * dataset.getColumnCount();
                if (dataset != null && dataset.getColumnCount() > 0) {
                        chart = ChartFactory.createBarChart(title1, title2, "票数", dataset,
                                        PlotOrientation.HORIZONTAL, false, true, false);
                        chart.addSubtitle(new TextTitle(subtitle));     //添加副标题
                }
        } else {                                                        //处理查看“各项所得票数”的请求
                dataset = getDataSetForBarAndOption(method);            //获取数据集
                title1 = "各项所得票数";
                title2 = "选项";
                width = 80 + 50 * dataset.getColumnCount();
                height = 400;
                if (dataset != null && dataset.getColumnCount() > 0) {
                        chart = ChartFactory.createBarChart3D(title1, title2, "票数",
                                dataset, PlotOrientation.VERTICAL, false, true, false);
                        chart.addSubtitle(new TextTitle(subtitle));     //添加副标题
                }
        }
        return chart;
}
```

（2）在 getChartForBar()方法中调用了 getDataSetForBarAndArea(String method)和 getDataSetForBar AndOption (String method)方法。其中，getDataSetForBarAndArea()方法用来创建按投票地区统计的柱形图的绘图数据集对象，其完整代码如下：

```
/**
 * @功能：创建用于绘制柱形图的绘图数据集对象（按投票地区统计）
 * @返回值：CategoryDataset 对象
*/
private CategoryDataset getDataSetForBarAndArea(String method) {
	DefaultCategoryDataset dataset = null;
	AreaDao areaDao = new AreaDao();
	List areas = null;
	if ("all".equals(method))                                  //按地区统计总投票数
		areas = areaDao.getAreas();
	else if ("day".equals(method))                             //按地区统计当日的投票数
		areas = areaDao.getAreasForDay();
	else if ("month".equals(method))                           //按地区统计当月的投票数
		areas = areaDao.getAreasForMonth();
	areaDao.closed();
	if (areas != null && areas.size() > 0) {
		dataset = new DefaultCategoryDataset();                //创建柱形图的绘图数据集对象
		for (int i = 0; i < areas.size(); i++) {
			AreaBean single = (AreaBean) areas.get(i);
			if (single.getAreaBallot() > 0)
				dataset.addValue(single.getAreaBallot(), "", single.getAreaName());  //添加绘图数据
		}
	}
	return dataset;
}
```

（3）getDataSetForBarAndOption()方法用来创建按投票项统计的柱形图的绘图数据集对象，其完整代码如下：

```
/**
 * @功能：创建用于绘制柱形图的绘图数据集对象（按投票项统计）
 * @返回值：CategoryDataset 对象
 */
private CategoryDataset getDataSetForBarAndOption(String method) {
	OptionDao optionDao = new OptionDao();
	List options = null;
	if ("all".equals(method))                                  //按投票项统计总得票数
		options = optionDao.getOptions();
	else if ("day".equals(method))                             //按投票项统计当日的得票数
		options = optionDao.getOptionsForDay();
	else if ("month".equals(method))                           //按投票项统计当月的得票数
		options = optionDao.getOptionsForMonth();
	optionDao.closed();
	DefaultCategoryDataset dataset = new DefaultCategoryDataset();
	for (int i = 0; i < options.size(); i++) {
		OptionBean single = (OptionBean) options.get(i);
		dataset.addValue(single.getOptionBallot(), "", single .getOptionName());   //添加绘图数据
	}
	return dataset;
}
```

心法领悟 582：解决按 ID 搜索字条时搜索结果不能正常显示的问题。

在实现按字条 ID 搜索字条时，遇到了这样一个问题：在没有嵌入界面时，该功能运行正常，但当嵌入界面后，搜索结果就不能正常显示了，一闪便不见了。仔细分析嵌入界面前与嵌入界面后的代码发现，在嵌入界面时，将原本用类型为 button 的<input>标签实现的“搜索”按钮替换成了类型为 image 的<input>标签，即将原来的普通按钮替换为图像域，由于图像域同“提交”按钮是等效的，所以当单击图像域后，表单将被提交，从而产生了搜索结果不能正常显示的现象。解决该问题的方法是，在图像域的 onclick 事件中添加以下代码：

```
return false;
```

这样该表单将不会被提交，搜索结果将正常显示。需要注意的是，上面的 return 语句一定要放在调用的搜

索函数 searchScrip()中。

实例 583　实现饼图统计功能

光盘位置：光盘\MR\23\583

中级

实用指数：★★★★☆

实例说明

饼图用来以百分比的形式展示一组相关数据之间的比例关系，此时这些数据将被看作一个整体。如图 23.17 所示展示了各项得票数占总得票数的百分比。

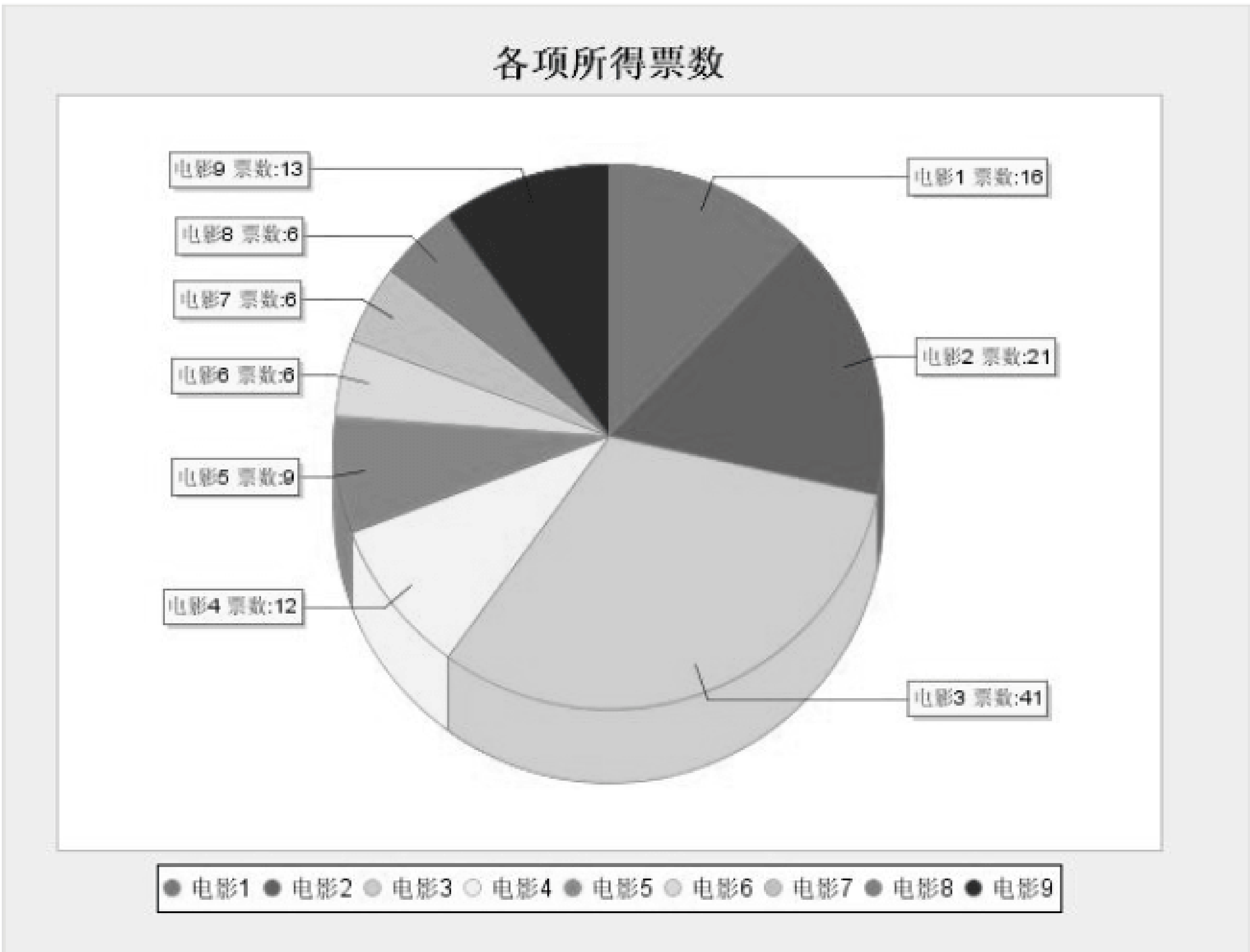


图 23.17　实现饼图统计功能

关键技术

本实例使用了 JFreeChar 绘制统计图。本书在第 7 章曾重点讲解了如何利用 JFreeChar 绘制各种图表，读者可参考。

（1）利用 VoteServlet 类中的 getChartForPie(String action, String method)方法创建饼形统计图对象，其完整代码如下：

```
/**
 * @功能：生成代表饼图的 JFreeChart 对象
 * @返回值：JFreeChart 对象
 */
private JFreeChart getChartForPie(String action, String method) {
        DefaultPieDataset dataset = null;
        JFreeChart chart = null;
        String title = "";
        String subtitle = "";
        width = 550;
        height = 430;
        //定义统计图的副标题
        if ("day".equals(method))
                subtitle = "一日统计(今日)";
        else if ("month".equals(method))
                subtitle = "一月统计(当前月)";
        if ("area".equals(action)) {                                    //处理查看“各省所投票数”的请求
                dataset = getDataSetForPieAndArea(method);               //获取数据集
```

```
            title = "各省所投票数图";
        } else {                                                    //处理查看“各项所得票数”的请求
            dataset = getDataSetForPieAndOption(method);             //获取数据集
            title = "各项所得票数";
        }
        if (dataset != null && dataset.getItemCount() > 0) {
            chart = ChartFactory.createPieChart3D(title, dataset, true, true, false);
            chart.addSubtitle(new TextTitle(subtitle));              //为统计图添加副标题
        }
        return chart;
}
```

（2）在 getChartForBar()方法中调用了 getDataSetForPieAndArea(String method)和 getDataSetForPie AndOption (String method)方法。getDataSetForPieAndArea()方法用来创建按投票地区统计的饼图的绘图数据集对象，其完整代码如下：

```
/**
 * @功能：创建用于绘制饼图的绘图数据集对象（按投票地区统计）
 * @返回值：DefaultPieDataset 对象
*/
private DefaultPieDataset getDataSetForPieAndArea(String method) {
        DefaultPieDataset dataset = null;
        AreaDao areaDao = new AreaDao();
        List areas = null;
        if ("all".equals(method))                                       //按地区统计总投票数
            areas = areaDao.getAreas();
        else if ("day".equals(method))                                  //按地区统计当日的投票数
            areas = areaDao.getAreasForDay();
        else if ("month".equals(method))                                //按地区统计当月的投票数
            areas = areaDao.getAreasForMonth();
        areaDao.closed();
        if (areas != null && areas.size() > 0) {
            dataset = new DefaultPieDataset();                          //创建饼图的绘图数据集对象
            for (int i = 0; i < areas.size(); i++) {
                AreaBean single = (AreaBean) areas.get(i);
                if (single.getAreaBallot() > 0)
                    dataset.setValue(single.getAreaName(), single.getAreaBallot());      //添加绘图数据
            }
        }
        return dataset;
}
```

注意： 在封装用来绘制饼图的绘图数据时，并不需要计算出百分比，直接传入绘图数据即可，饼图的绘图数据集会自动计算出该数据占传入数据总和的百分比。

（3）getDataSetForPieAndOption()方法用来创建按投票项统计的饼图的绘图数据集对象，其完整代码如下：

```
/**
 * @功能：创建用于绘制饼图的绘图数据集对象（按投票项统计）
 * @返回值：DefaultPieDataset 对象
*/
private getDataSetForPieAndOption(String method) {
        DefaultPieDataset dataset = null;
        OptionDao optionDao = new OptionDao();
        List options = null;
        if ("all".equals(method))                                       //按投票项统计总得票数
            options = optionDao.getOptions();
        else if ("day".equals(method))                                  //按投票项统计当日的得票数
            options = optionDao.getOptionsForDay();
        else if ("month".equals(method))                                //按投票项统计当月的得票数
            options = optionDao.getOptionsForMonth();
        optionDao.closed();
        if (options != null && options.size() != 0) {
            dataset = new DefaultPieDataset();                          //创建饼图的绘图数据集对象
            for (int i = 0; i < options.size(); i++) {
                OptionBean single = (OptionBean) options.get(i);
                if (single.getOptionBallot() > 0)
                    dataset.setValue(single.getOptionName(), single.getOptionBallot()); //添加绘图数据
```

```
            }
        }
        return dataset;
}
```

秘笈心法

心法领悟 583：在 IE 浏览器中禁用 Cookie。

如果在 IE 浏览器中设置了禁用 Cookie 功能，但访问 URL 中的主机名部分为本地回路地址 12.0.0.1 或是被解析成这个地址的主机名（例如，localhost），则 IE 浏览器仍然能够接受 Web 服务器发送的 Cookie 信息，并向 Web 服务器回传 Cookie 信息。

实例 584	双击鼠标展开图片 光盘位置：光盘\MR\23\584	中级 实用指数：★★★☆

在投票系统中，提供了查询统计结果功能。统计结果初始状态是包含在一个带有滚动条的页面中，如图 23.18 所示；当用户在统计结果中双击鼠标，即可将图片展开，如图 23.19 所示。

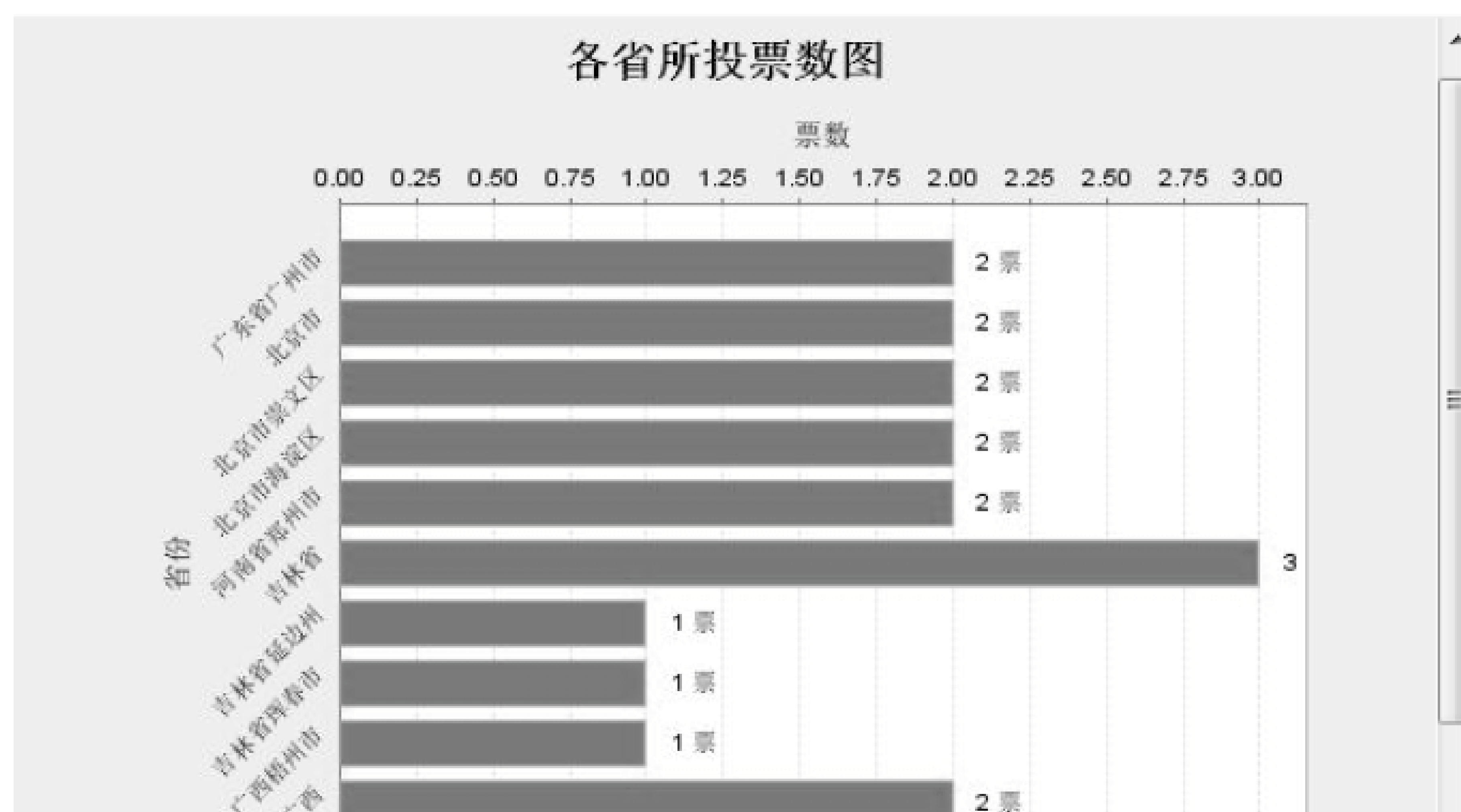


图 23.18　统计结果的初始状态

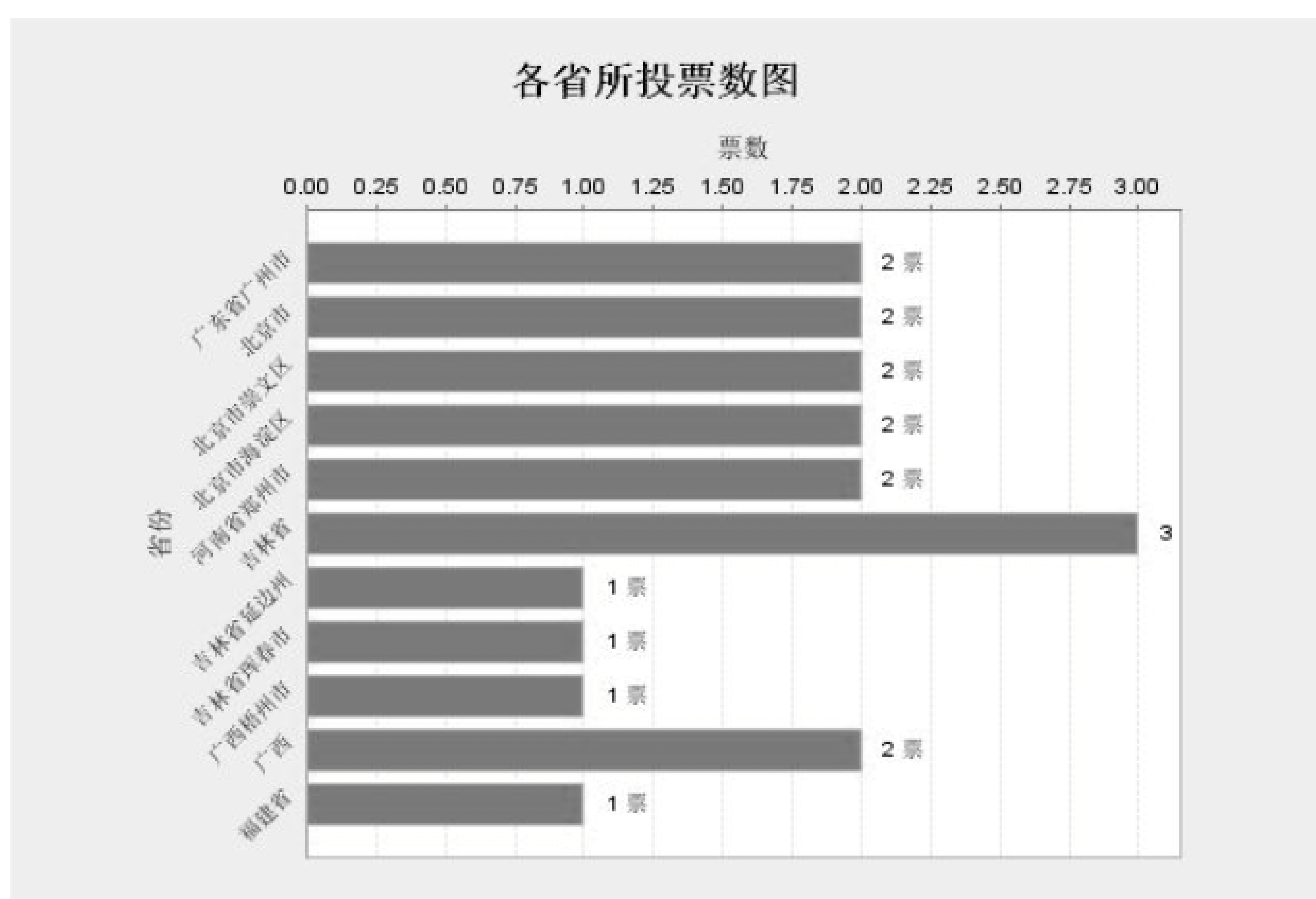


图 23.19　统计结果展开后的状态

关键技术

本实例中将统计结果在 div 层显示，并在该层中添加鼠标事件。

设计过程

（1）在 showresult.jsp 页中，将图片嵌入到 div 层中。具体的嵌入代码如下：

```
<div ondblclick="size()">
	<img id="pic" src="plot/${requestScope.path}.jpg" title="双击收缩图片"
					alt="正在加载图片，请稍等..." usemap="#mymap" style="border:0">
</div>
```

（2）为 div 元素设置 ondblclick 属性，表示当双击该 div 层时将执行脚本方法 size()。脚本方法 size()用来实现缩放图片，其完整代码如下：

```
var mark1="off";
var mark2="off";
function size(){
	//获取父页面（toresult.jsp）中 id 属性值为 resultpic 的元素（这里为 iFrame 框架）
	var tag1=parent.document.getElementById("resultpic");
	if(mark1=="off"){
		mark1="on";
		//将 tag1 元素的高度设置为 showresult.jsp 页面的高度，实现放大效果
		tag1.height=document.body.scrollHeight;
	}else{
		mark1="off";
		//将 tag1 元素的高度设置为指定值，实现缩小效果
		tag1.height=350;
	}
	//获取父页面的父页面（main.jsp）中 id 属性值为 resultpage 的元素（这里为 iFrame 框架）
	var tag2=parent.parent.document.getElementById("resultpage");
	if(mark2=="off"){
		mark2="on";
		//将 tag2 元素的高度设置为 showresult.jsp 的父页面 toresult.jsp 的高度，实现放大效果
		tag2.height=parent.document.body.scrollHeight;
	}else{
		mark2="off";
		//将 tag2 元素的高度设置为指定值，实现缩小效果
		tag2.height=450;
	}
}
```

秘笈心法

心法领悟 584：如何快速入门新技术。

对于软件开发人员来说，掌握新技术是非常重要的，否则很快便会落后，给日常工作带来极大的问题。其实，任何一门新技术的开发厂商都会提供一些针对这门技术的应用示例，其中会包含一个最简单的入门示例，初学者只要找到入门示例，然后参照其源代码和配置信息，自己编写一个同样简单的程序，即可轻松、快速地掌握该程序的编译、配置和运行等完整的过程。

23.4　B2C 电子商务网站

实例 585	添加商品到购物车 光盘位置：光盘\MR\23\585	中级 实用指数：★★★

实例说明

在电子商务网站中，将商品添加到购物车功能是非常重要的。要实现这一功能，首先要将商品的信息在页

面中显示，之后提供给用户“放入购物车”与“查看购物车”超链接。本实例的运行结果如图 23.20 所示。



图 23.20　添加商品到购物车

关键技术

在 iFrame 框架中显示的内容独立于主页中的内容，即对 iFrame 框架中的内容进行操作时不会影响 iFrame 框架外的内容。在播客系统中播放某个视频时即可应用 iFrame 显示视频的评论，这样在对评论进行分页查看时不会影响视频的播放。本系统应用 iFrame 框架实现了某版块下非置顶主题的列表显示和查看某主题时回复帖的列表显示等。

iFrame 框架的使用非常简单，该方法如下：

```
<iframe   id=""                                    //框架的 ID
          name=""                                  //框架的名称
          src=""                                   //框架包含的资源文件路径
          width=""                                 //框架宽度
          height=""                                //框架高度
          frameborder="0"                          //框架边框宽度
          scrolling="no">                          //是否带有滚动条
</iframe>
```

其中，src 属性可以指定一个具体的物理路径，例如 a/b/c.jsp，也可以指定一个映射路径，例如 a/b/c。scrolling 属性用来设置框架是否带有滚动条。该属性具有 3 个属性值：no 表示不带有滚动条；yes 表示带有滚动条；auto 表示将根据包含的内容及框架的宽度与高度值自动显示滚动条。

iFrame 框架的高度是不能根据所包含的内容自动进行调整的，可以通过 JavaScript 脚本实现。例如，有 a.jsp 和 b.jsp 页面，在 a.jsp 中通过 iFrame 包含 b.jsp，关键代码如下：

```
<center>这是 a.jsp 页面！</center>
<hr>
<iframe id="aa" name="AA" src="b.jsp" width="100%" height="0" frameborder="0" scrolling="no">
</iframe>
```

创建 ShopcarDao 类，用来实现基于数据库的针对购物车的各种操作。在该类中定义了一个类型为 DB 的属性 mydb，通过该属性连接及操作数据库。addBuyNum()和 addBuyGoods()方法的具体代码如下：

```
public int addBuyNum(Object[] params){
        String sql="update tb_shopcar set shopcar_buygoodscount=shopcar_buygoodscount+1 where shopcar_id=? and shopcar_buygoodsid=?";
        return getUpdate(sql,params);
}
public int addBuyGoods(Object[] params){
        String sql="insert into tb_shopcar values(?,?,?)";
        return getUpdate(sql,params);
}
```

因为都是更新数据库的操作，所以除 SQL 语句不同之外，其他操作都是相同的。将这些相同的部分在

getUpdate()私有方法中完成，其关键代码如下。

```
mydb.doPstm(sql, params);
int i=-1;
i=mydb.getCount();
return i;
```

秘笈心法

心法领悟 585：创建临时表保存系统。

在设计电子商务网站中的购物车时，一定要包含对购物车中的信息进行管理的功能。例如，购物车中的信息只能保留 3 天，超过 3 天后的记录就要被删除，这里可以创建一个临时表来保存购物车创建的时间。

实例 586	查看购物车 光盘位置：光盘\MR\23\586	中级 实用指数：★★★

实例说明

将商品放入购物车后，用户可以通过单击“查看购物车”超链接查看购物车中的商品。如果购物车中有商品，将显示商品信息。此外，还可以对购物车中的商品进行删除。本实例运行结果如图 23.21 所示。



图 23.21　查看购物车

由于购物车中可能会包含多条记录，所以在显示购物车内容的页面中，需要通过循环来显示内容。本实例中使用 JSTL 标签中的<c:forEach>循环标签来显示，其语法如下。

语法 1：数字索引迭代。

```
<c:forEach begin="start" end="finish" [var="name"] [varStatus="statusName"]>
[step="step"]
        标签主体
</c:forEach>
```

其中，begin 和 end 属性是必选的属性，其他属性均为可选属性。

语法 2：集合成员迭代。

```
<c:forEach items="data" [var="name"] [begin="start"] [end="finish"] [step="step"]
[varStatus="statusName"]>
        标签主体
</c:forEach>
```

其中，items 属性是必选属性，通常使用 EL 表达式指定；其他属性均为可选属性。

<c:forEach>标签的各属性的详细介绍如表 23.1 所示。

表 23.1　<c:forEach>标签的属性

属　　性	类　　型	描　　述	引用 EL
items	数组、集合类、字符串和枚举类型	被循环遍历的对象，多用于数组与集合类	可以
var	String	循环体的变量，用于存储 items 指定的对象的成员	不可以
begin	int	循环的起始位置，如果没有指定，则从集合的第一个值开始迭代	可以
end	int	循环的终止位置，如果没有指定，则一直迭代到集合的最后一位	可以
step	int	循环的步长	可以
varStatus	String	循环的状态信息，可以取 index、count、first、last 等值	不可以

（1）创建 ShopcarDao 类，在该类中创建 getShopcar()方法，在该方法中获取符合条件的购物车信息。具体代码如下：

```
ShopcarBean =new ShopcarBean();                              //创建购物车 JavaBean 实例
shopcar.setShopcarId(shopcarid);                             //存储购物车 ID
String sql="select * from tb_shopcar where shopcar_id=? and shopcar_buygoodscount!=0 order by id desc";
Object[] params={shopcarid};
mydb.doPstm(sql, params);                                    //查询数据库，mydb 为 DB 类的实例
ResultSet rs = mydb.getRs();                                 //获取结果集
GoodsDao goodsDao=new GoodsDao();
while(rs.next())                                             //依次查询出每个商品信息并保存到购物车 JavaBean 的 List 集合中
        shopcar.setShopcarBuyGoodss(getBuyGoodsToShopcar(goodsDao,rs.getInt(3),rs.getInt(4)));
return shopcar;
```

（2）创建显示购物车的 JSP 页面，在该页面中定义 Form 表单，该表单将被提交给 ShopcarServlet 中的 submitDispatcher()方法，然后在该表单中遍历存储购物车信息的 JavaBean 实例中的商品列表，输出商品信息。其关键代码如下：

```
<c:set var="myshopcar" value="${requestScope.shopcar}"/>
<c:if test="${(empty myshopcar) or (empty myshopcar.shopcarBuyGoodss)}">
<tr><td colspan="6" align="center">您还没有挑选商品到购物车中。</td></tr></c:if>
<c:if test="${(!empty myshopcar) and (!empty myshopcar.shopcarBuyGoodss)}">
<form action="submitshopcar" name="updateform" method="post">
<c:forEach var="buygoods" varStatus="bvs" items="${myshopcar.shopcarBuyGoodss}">
        <c:if test="${!empty buygoods}">
      <input type="hidden" name="buygoodsids" value="${buygoods.id}">                //商品 ID
<input type="hidden" name="buygoodsstorenum" value="${buygoods.goodsStoreNum}">      //库存
      <tr>
        <td>${bvs.count}</td>                                                        //输出序号
        <td>${buygoods.goodsName}</td>                                               //输出商品名称
        <td>￥${buygoods.goodsPrice}</td>
        <td>
                    //以文本框显示购买数量
                    <input type="text" name="buygoodsnum" value="${buygoods.goodsBuyNum}"><br>
                    <font color="red">${requestScope.messages[bvs.index]}</font>     //提示信息
            </td>
        <td>￥${buygoods.goodsMoney}</td>                                            //商品单价
        <td><a href="remove?goodsId=${buygoods.id}">删除</a></td>
      </tr>
      <c:set var="totalmoney" value="${totalmoney+buygoods.goodsMoney}"/>            //计算商品总价格
        </c:if>
        </c:forEach>
        <input type="hidden" name="goodsprices" value="${totalmoney}">
        <tr><td colspan="6"><hr></td></tr>
      <tr><td colspan="6">总金额：<input type="text" name="goodsprices" value="${totalmoney}" style="border:0" disabled></td></tr>
        <tr>
```

```
        <td colspan="3">
            <input type="submit" name="whichsubmit" value="修改数量">
            <a href="clearshopcar">清空购物车</a>
        </td>
        <td colspan="3">
            <%
                Object loginer=session.getAttribute("loginer");
                if(loginer==null||!(loginer instanceof com.valuebean.UserBean))
                    out.print("您没有登录，不能进行");
                else
                    out.println("您已经登录，可以进行");
            %>
            <input type="submit" value="商品结算">
        </td>
    </tr>
</form>
</c:if>
```

秘笈心法

心法领悟 586：在页面中定义 JSTL 标签库的位置。

要在页面中使用 JSTL 标签，不仅要求在项目中包含 jstl.jar 文件，还要在页面的首行中使用<%@ taglib%>指令定义标签库的位置和访问前缀。例如，使用核心标签库的 taglib 指令格式如下：

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

实例 587　修改商品数量

光盘位置：光盘\MR\23\587

中级

实用指数：★★☆

实例说明

在如图 23.21 所示的购物车页面中，修改商品的数量，然后单击“修改数量”按钮，可实现商品数量的批量修改。修改完成后将重新发起“查看购物车”的请求，返回到购物车页面后就会看到，修改成功的商品将会显示“修改成功”提示信息，否则显示“库存不足”提示信息，结果如图 23.22 所示。

图 23.22　修改商品数量

关键技术

本实例实现的是批量修改购物车中的商品数量，用户可以在页面中给出的“数量”文本框中设置，之后系统将修改数据表中的数据。

创建 ShopcarServlet 类，在该类中定义处理“修改数量”的方法 shopcar_validateBuyNum()。在该方法中，将获取购物车表单中的所有商品 ID 和库存量以及文本框中的购买数量，然后依次进行判断。该方法的关键代码如下：

```
boolean mark=true;
//获取购物车表单中的所有商品库存量
String[] goodsStoreNums=request.getParameterValues("buygoodsstorenum");
String[] buyNums=request.getParameterValues("buygoodsnum");             //获取所有商品的购买数量
String[] goodsIds=request.getParameterValues("buygoodsids");            //获取所有商品的 ID
String shopcarid=seeshopcarcookie(request,response);                    //获取购物车 ID
if(goodsIds!=null&&goodsIds.length!=0&&shopcarid!=null){
        Map messages=new HashMap();
```

```
        Object[] params=new Object[3];
        ShopcarDao shopcarDao=new ShopcarDao();
        for(int i=0;i<goodsIds.length;i++){                          //遍历购物车中的商品
            int int_buyNum=Integer.parseInt(buyNums[i]);              //获取当前商品的购买数量
            int int_goodsStoreNums=Integer.parseInt(goodsStoreNums[i]);//获取当前商品的库存量
            if(int_buyNum>int_goodsStoreNums){                        //库存不足
                mark=false;
                messages.put(i,"⊥库存不足！ ");
            }
            else if(int_buyNum<=0)                                    //如果将购买数量修改为 0 或负数，从 tb_shopcar 表中删除该记录
                shopcarDao.deleteGoods(shopcarid, Integer.parseInt(goodsIds[i]));
            else{                                                     //库存足够
                params[0]=int_buyNum;
                params[1]=shopcarid;
                params[2]=goodsIds[i];
                shopcarDao.updateBuyNum(params);                      //操作数据库修改商品购买数量
                messages.put(i,"√修改成功！ ");
            }
        }
        request.setAttribute("messages",messages);
        shopcarDao.closed();
}
else
 mark=false;
return mark;
```

秘笈心法

心法领悟 587：不要使用 GET 方式提交包含口令的 FORM 表单。

千万不要使用 GET 方式提交访问口令的表单，否则访问者输入的口令将作为 URL 的一部分存储在多个地方，其中包括 Web 浏览器的历史记录文件和 Web 服务器日志文件。

实例 588	生成订单 光盘位置：光盘\MR\23\588	中级 实用指数：★★★★☆

要为选购的商品进行结算，先要生成一个订单，订单信息中包括收货人信息、送货方式、支付方式、购买的商品以及订单总价格等，生成订单页面的运行结果如图 23.23 所示。

图 23.23　生成订单

关键技术

本实例中购物车中的商品可以被保存 3 天，实现该功能主要是应用 Cookie 和数据表。一个数据表用来存储用户购物车中的商品，另一个用来存储购物车的创建时间；Cookie 用来存储用户购物车 ID。在显示购物车中的商品时，首先从 Cookie 中获取购物车 ID，然后根据该 ID 查询其中一个数据表获取购物车中的商品，最后显示到页面中。这样可能会存在一些不同步的问题。例如，用户将商品放入购物车后，向 Cookie 中写入购物车 ID，并向数据表中记录该购物车的创建时间。如果用户删除了自己计算机中的 Cookie，那么数据表中存储的相应记录就成为无效记录（这些记录可通过作业进行清理）；但如果数据表中的记录先被删除，则将导致从 Cookie 中获取的购物车 ID 在数据表中没有对应的记录。在这两种情况下，本系统都会重新为用户生成一个购物车 ID，然后写入 Cookie 中，同时向数据表中记录该购物车的创建时间。具体实现如 ShopcarServlet 的 buy()方法中的一段代码：

```
Date now=new Date();                                                    //获取当前时间
TempDao tempDao=new TempDao();
//查询客户端 Cookie 中是否保存了一个购物车 ID 值
String shopcarid=seeshopcarcookie(request,response);
if(shopcarid==null||shopcarid.equals("")||!tempDao.isexist(shopcarid)){   //没有保存
        //生成一个购物车 ID 保存到客户端 Cookie 中，并返回该 ID 值
        shopcarid=addshopcarcookie(request,response,now);
        //记录该购物车 ID 和创建时间到数据表中
        tempDao.saveShopcarCreateTime(shopcarid, StringHandler.timeTostr(now));
}
```

（1）在 ShopcarServlet 中创建 payforMoney()方法，该方法首先验证库存量，若库存不足，则重新获取购物车中的商品并返回购物车页面；若库存足够，则将购物车表单中的所有商品 ID 和对应的购买数量转换为以“,”分隔的字符串并保存，然后将请求转发到填写表单信息的 fillOrderform.jsp 页面。具体代码如下：

```
if(shopcar_validateBuyNum(request, response)){
        String buygoodsids=StringHandler.ArrayToString(request.getParameterValues("buygoodsids"));
        String buygoodsnum=StringHandler.ArrayToString(request.getParameterValues("buygoodsnum"));
        request.setAttribute("buygoodsids", buygoodsids);
        request.setAttribute("buygoodsnum", buygoodsnum);
        RequestDispatcher rd=request.getRequestDispatcher("/fillOrderform.jsp");
        rd.forward(request,response);
}
else
        showshopcar(request,response);
```

（2）创建填写表单信息的 JSP 页面，在该页面中实现各信息的输入。关键代码如下：

```
<form action="createorderform" name="orderform" method="post">
  <input type="hidden" name="buygoodsids" value="${requestScope.buygoodsids}">
  <input type="hidden" name="buygoodsnum" value="${requestScope.buygoodsnum}">
  <input type="hidden" name="goodsprices" value="${param.goodsprices}">
  <table>
      <tr><td colspan="3">收货人信息</td></tr>
      <tr><td>收  货  人：</td><td colspan="2"><input type="text" name="getter" size="30" ></td></tr>
    ……//省略了收货人其他信息
    <tr><td colspan="3">送货方式</td></tr>
    <tr>
      <td><input type="radio" id="shipment1" name="shipment" value="1" >普通快递送货上门</td>
      <td colspan="2"> <div id="shipmenttimes" style="display:'none'">
       送货时间：<select name="shipmenttime" onchange="shipmentmessage.innerHTML=''">
                        <option value="">-请选择时间-</option>
                        <option value="1">不限时间</option>
                        <option value="2">3 天内</option>
                        <option value="3">1 周内</option>
                        <option value="4">1 月内</option>
                 </select>（支持货到付款）【运费：20 元】</div></td>
     </tr>
    ……//省略了其他送货方式
     <tr><td colspan="3">支付方</td></tr>
     <tr>
```

```
        <td colspan="3"><input type="radio" id="payment1" name="payment" value="1">网上支付<br>
        <div id="networkpayments" style="padding-left:30;display:">
        <input type="radio" id="networkpayment1" name="networkpayment" value="1">工商银行<br>
        <input type="radio" id="networkpayment4" name="networkpayment" value="4">支付宝支付<br>
        ……//省略其他网上支付方式
        </div></td>
       </tr>
       ……//省略其他支付方式
    </table>
</form>
```

（3）在 ShopcarServlet 中创建 createorderform()方法，获取表单中填写的订单信息；然后保存一份到数据库中，另一份保存到存储订单信息的 OrderformBean 中以便在 JSP 页面中输出订单信息；最后依次修改订单中商品的库存量并清空购物车。createorderform()方法的关键代码如下：

```
String buygoodsids=request.getParameter("buygoodsids");
String buygoodsnum=request.getParameter("buygoodsnum");
int loginerid=((UserBean)request.getSession().getAttribute("loginer")).getId();
String shipment=request.getParameter("shipment");
String shipmenttime=request.getParameter("shipmenttime");
String payment=request.getParameter("payment");
String networkpayment=request.getParameter("networkpayment");
……//省略获取其他订单信息的代码
/** 计算订单总价格：根据选择的送货方式加入运费 */
float totalprices=0;
if(shipment.equals("1"))
        totalprices=20+goodsprices;
else if(shipment.equals("2"))
        totalprices=30+goodsprices;
else if(shipment.equals("3"))
        totalprices=40+goodsprices;
request.setAttribute("goodsprices",goodsprices);
request.setAttribute("totalprices",totalprices);
Object[] params={ loginerid,getter,address,postalcode,linkphone,shipment,shipmenttime,
payment,networkpayment,totalprices,time,status,buygoodsids,buygoodsnum};
OrderformDao orderformDao=new OrderformDao();
int i=orderformDao.addOrderform(params);                              //保存订单信息到数据表中
if(i<=0){     ……//省略生成错误提示信息及转发到提示页面的代码}
else{
        int orderformnumber=orderformDao.getOrderformNumber(loginerid, time);          //获取订单编号
        List buygoodslist=orderformDao.getBuyGoodsToOrderform(orderformnumber);        //获取订单中商品
        OrderformBean orderform=new OrderformBean();
        orderform.setOrderformNumber(orderformnumber);
        orderform.setOrderformBuyGoods(buygoodslist);
        ……//省略保存订单其他信息的代码
        request.setAttribute("orderform", orderform);                               //保存订单到 request 中
        /* 修改商品库存量 */
        String[] goodsids=buygoodsids.split(",");
        String[] goodsnum=buygoodsnum.split(",");
        GoodsDao goodsDao=new GoodsDao();
        for(int k=0;k<goodsids.length;k++)
                goodsDao.updateStoreNum(Integer.parseInt(goodsnum[k]),Integer.parseInt(goodsids[k]));
            goodsDao.closed();
        /* 清空购物车 */
        deleteshopcarcookie(request,response);                                      //删除存储购物车 ID 的 Cookie
        String shopcarid=seeshopcarcookie(request,response);
        TempDao tempDao=new TempDao();
        tempDao.deleteShopcar(shopcarid);                                           //删除 tb_temp 表中保存的购物车创建信息
}
```

心法领悟 588：用时间间隔限制用户连续提交。

使用验证码不仅可以保护系统的安全，还可以防止用户重复提交。不过，这种方式不能有效地阻止乱采用手工方式连续发送垃圾信息的行为。因此，一些网站通常在 Session 中记录用户发帖的时间，然后通过一个时间间隔来限制用户连续发帖的数量。

23.5　在　线　音　乐

实例 589　试听歌曲并同步显示歌词

光盘位置：光盘\MR\23\589

中级

实用指数：★★★★☆

实例说明

现在，在网络上听音乐可以说是一种时尚，因此很多网站都提供了大量的歌曲，访客可以听歌，也可以下载。本实例调用客户端的 Windows Media Player 播放器，实现试听歌曲并同步显示歌词，如图 23.24 所示。

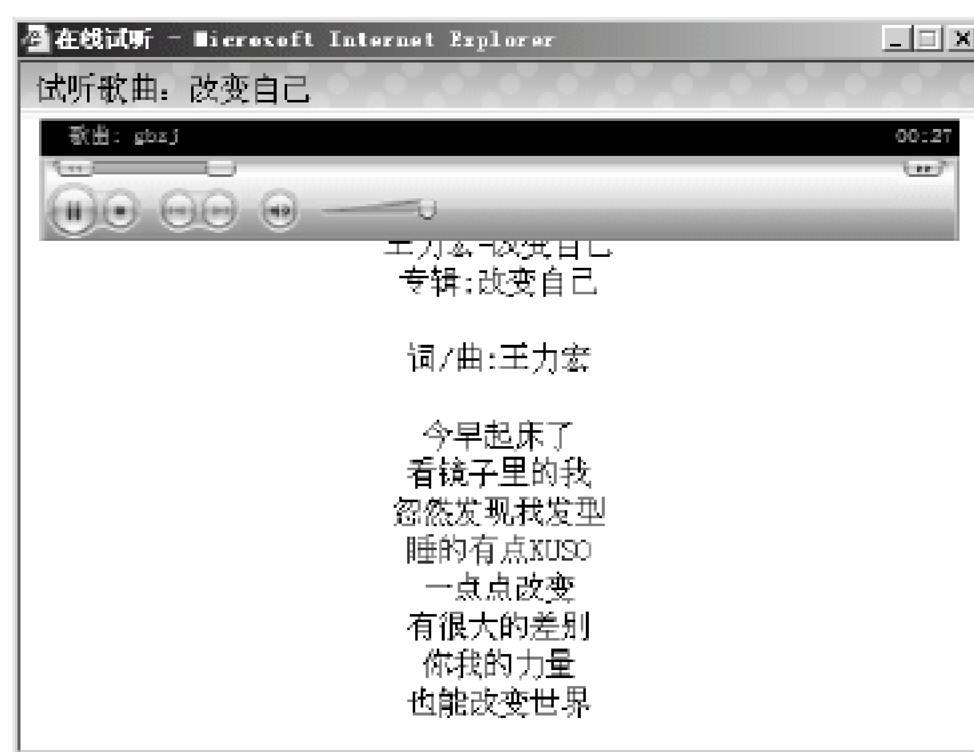


图 23.24　试听歌曲并同步显示歌词

关键技术

想要通过网页在客户端播放音乐或视频，客户端必须安装指定的播放器。为了当客户端机器上没有安装指定的播放器时给予相关提示，可以在程序中添加检测客户端是否安装指定播放器的功能。

目前比较常用的两种播放器是 Windows Media Player 和 RealPlayer，下面将分别介绍验证客户端是否安装这两种播放器的方法。

检测客户端是否安装 Windows Media Player 播放器的具体步骤如下：

（1）编写自定义的 JavaScript 函数 isInsalled()，具体代码如下：

```
function isInsalled(oID){
	return returnValue; returnValue=document.getElementById("temp").isComponentInstalled(oID,"componentid")?true:false;
}
```

（2）在页面的<body></body>标签中，添加一个名为 temp 的<span>标签，代码如下：

```
<span style="behavior:url(#default#clientCaps)" id="temp"></span>
```

（3）调用步骤（1）中的自定义函数检测客户端是否安装了 Windows Media Player 播放器，如果客户端已经安装 Windows Media Player 播放器，将返回 true，否则返回 false。具体代码如下：

```
isInsalled('{22d6f312-b0f6-11d0-94ab-0080c74c7e95}')
```

其中，{22d6f312-b0f6-11d0-94ab-0080c74c7e95}为 Windows Media Player 播放器 ClassID。

（1）在歌曲信息相关的 Action 实现类中，编写实现在线试听的方法 tryListen()。在该方法中，首先获取要播放歌曲的 ID，并根据该 ID 调用 SongDAO 类中的 tryListen()方法获取该歌曲的信息，然后读取该歌曲对应的歌词文件（即 LRC 文件），将读取的歌词内容连接成一个字符串，并统计歌词的行数，再将歌词的行数、歌词内容、当前页的歌曲信息和当前试听的歌曲名称保存到 HttpServletRequest 对象中，最后将页面重定向到在线试听页面。tryListen()方法的具体代码如下：

```
public ActionForward tryListen(ActionMapping mapping, ActionForm form,
	HttpServletRequest request, HttpServletResponse response) {
	int id = Integer.parseInt(request.getParameter("id"));
	//声明一个数组，该数组的第一个元素为歌曲名称，第二个元素为歌曲的文件名
	String[] urlAndName = songDAO.tryListen(id);                    //根据歌曲 ID 获取歌曲信息
		/** **************获取歌词************************* */
		String lrcRealPath = request.getRealPath("/");
		String mp3RealPath = lrcRealPath.substring(0, lrcRealPath
				.lastIndexOf("/") + 1)+ "music/" + urlAndName[1];
		request.setAttribute("realPath", mp3RealPath);          //保存要播放歌曲的完整路径
		lrcRealPath = lrcRealPath+ "music/"+
			urlAndName[1].substring(0, urlAndName[1].lastIndexOf(".") + 1)+ "lrc";   // LRC 文件路径
		File lrcFile = new File(lrcRealPath);
```

```
        songDAO.holdoutAdd(id);                                //将试听次数加 1
        String content = "";
        int lineNumber = 0;
        if (lrcFile.exists()) {
            FileInputStream lrcf;
            try {
                lrcf = new FileInputStream(lrcRealPath);
                int rs = 0;
                //available()方法可以不受阻塞地从此输入流中读取（或跳过）的估计剩余字节数
                byte[] data = new byte[lrcf.available()];
                while ((rs = lrcf.read(data)) > 0) {
                    content += new String(data, 0, rs);        //将歌词内容连接为一个字符串
                }
                //分析字符串中共包括多少个中括号对“[]”
                StringTokenizer st = new StringTokenizer(content, "\\[*\\]");
                lineNumber = st.countTokens();                 //返回分析的结果
            } catch (Exception e) {
                e.printStackTrace();
            }
    }
        /** ********************************************** */
        request.setAttribute("lineNumber", lineNumber);        //保存歌词的行数
        request.setAttribute("lrcContent", content);           //保存歌词内容
        request.setAttribute("fileURL", urlAndName[1]);        //保存当前页的歌曲信息
        request.setAttribute("songName", urlAndName[0]);       //保存当前试听的歌曲名称
        return mapping.findForward("tryListen");               //重定向页面到在线试听页面
    }
```

（2）在在线试听页面中，实现播放歌曲并设置歌词同步显示，定义解析 LRC 歌词的函数。具体代码如下：

```
function lrcClass(lyric){            //参数为歌词内容
        this.inr = [];
        this.oTime = 0;
        this.dts = -1;
        this.dte = -1;
        this.dlt = -1;
        this.ddh;
        this.fjh;
        //获取歌词中是否有时间补偿值（单位为毫秒），正数表示整体提前，负数表示整体滞后
        if(/\[offset\:(\-?\d+)\]/i.test(lyric)) this.oTime = RegExp.$1/1000;
        lyric = lyric.replace(/\[\:\][^$\n]*(\n|$)/g,"$1");
        lyric = lyric.replace(/\[[^\[\]\:]*\]/g,"");
        lyric = lyric.replace(/\[[^\[\]]*[^\[\]\d]+[^\[\]]*\:[^\[\]]*\]/g,"");
        lyric = lyric.replace(/\[[^\[\]]*\:[^\[\]]*[^\[\]\d\.]+[^\[\]]*\]/g,"");
        while(/\[[^\[\]]+\:[^\[\]]+\]/.test(lyric)){
            lyric = lyric.replace(/((\[[^\[\]]+\:[^\[\]]+\])+[^\[\r\n]*)[^\[]*/,"\n");
            var zzzt = RegExp.$1;
            /^(.+\])([^\]]*)$/.exec(zzzt);
            var ltxt = RegExp.$2;
            var eft = RegExp.$1.slice(1,-1).split("][");
            for(var ii=0; ii<eft.length; ii++){
                var sf = eft[ii].split(":");
                var tse = parseInt(sf[0],10) * 60 + parseFloat(sf[1]);
                var sso = { t:[] , w:[] , n:ltxt }
                sso.t[0] = tse-this.oTime;
                this.inr[this.inr.length] = sso;
            }
        }
        this.inr = this.inr.sort( function(a,b){return a.t[0]-b.t[0];} );
        for(var ii=0; ii<this.inr.length; ii++){
            while(/<[^<>]+\:[^<>]+>/.test(this.inr[ii].n)){
                this.inr[ii].n = this.inr[ii].n.replace(/<(\d+)\:([\d\.]+)>/,"%=%");
                var tse = parseInt(RegExp.$1,10) * 60 + parseFloat(RegExp.$2);
                this.inr[ii].t[this.inr[ii].t.length] = tse-this.oTime;
            }
            lrcLine_will1.innerHTML = "<font>"+ this.inr[ii].n.replace(/&/g,"&").replace(/</g,"<").replace(/>/g,">").replace(/%=%/g,"</font>
                <font>") +" </font>";
            var fall = lrcLine_will1.getElementsByTagName("font");
            for(var wi=0; wi<fall.length; wi++){
                this.inr[ii].w[this.inr[ii].w.length] = fall[wi].offsetWidth;
```

```
        }
        this.inr[ii].n = lrcLine_will1.innerText;
    }
```

秘笈心法

心法领悟 589：LRC 歌词格式。

LRC 歌词是具备一定的格式的，[MM:SS.MS]用于指定时间；[ar:演唱者名]用于指定演唱者；[ti:歌曲名]用于指定歌曲名；[al:专辑名]用于指定专辑名；[by:歌词编辑者]用于指定歌词编辑者；[Offset:MS]用于调整整个歌词文件的时间标签值，可以是负值（也是 LRC 歌词格式中，唯一可以使用负值的时间标签），单位是毫秒。

实例 590	添加歌曲 光盘位置：光盘\MR\23\590	中级 实用指数：★★★★☆

实例说明

为了方便用户对在线音乐网站进行管理，还需要为其加入添加歌曲的功能。例如，在本实例的甜橙音乐网中，管理员登录网站后台后，单击“添加歌曲”超链接，即可打开添加歌曲页面，如图 23.25 所示。默认情况下，该页面中的两个“上传文件”按钮都是不可用的。当选择歌曲类别、添加歌曲名、演唱者后，单击“检测该歌曲是否上传”按钮，如果该歌曲没有上传，则歌曲文件后面的“上传文件”按钮可用，否则将给出提示。当上传歌曲文件成功后，歌词文件后面的“上传文件”按钮也可用，这时即可上传歌词文件。歌曲信息添加完成后，单击“保存”按钮，即可将该歌曲添加到服务器中。

图 23.25 添加歌曲

关键技术

实现添加歌曲是网站后台的功能，首先需要将歌曲上传到服务器上，本实例应用 jspSmardUpload 组件实现文件的上传。为了实现歌词同步显示，需要保证歌词文件和歌曲文件同名（例如，歌曲文件名为 gbzj.mp3，歌词文件的名称就应该是 gbzj.lrc）。这样，在上传歌词文件时，就需要将歌词文件重命名。在应用 jspSmartUpload 组件上传文件时，可以通过文件上传组件的 getFile()方法获取 Files 类的对象，然后通过 Files 类的 getFile()方法获取文件集合中指定的文件对象，再通过该文件对象的 saveAs()方法将文件进行重命名上传即可。关键代码如下：

```
upFile.getFiles().getFile(0).saveAs("/music/"+fileName);
```

（1）编写上传文件的代码，将选择的歌曲文件上传到服务器中该程序所在文件夹下的 music 文件夹中，并将该文件重命名。具体代码如下：

```
<%@ page contentType="text/html; charset=gb2312" language="java" import="java.text.*,java.util.*"%>
<jsp:useBean id="upFile" scope="page" class="com.jspsmart.upload.SmartUpload" />
<%
upFile.initialize(pageContext);                                    //初始化文件上传下载组件
upFile.upload();
//格式化文件大小为两位小数
String fileSize=new DecimalFormat("#.##").format(upFile.getFiles().getSize()/1024.00/1024.00)+"M";
if(upFile.getFiles().getSize()>5000000){                           //检测上传文件是否过大
 out.println("<script>alert('您上传的文件太大，不能完成上传！');history.back(-1);</script>");
}else{
        String format=upFile.getFiles().getFile(0).getFileExt();   //获取文件的扩展名，但不包括“.”号
```

```
		Calendar ca=Calendar.getInstance();
		String fileName=String.valueOf(ca.getTimeInMillis())+"."+format;			//重新生成文件名
		if("mp3".equals(format) || "wma".equals(format)){					//判断格式是否合法
		//将上传后的文件名、文件大小和文件类型设置到添加歌曲页面的相应表单元素中，并设置上传歌词的按钮可用
		out.println("<script>opener.form1.fileURL.value='"+fileName+"';opener.form1.fileSize.value='"+fileSize+"';opener.form1.format.value='"+format+
"';opener.form1.lrcUp.disabled='';window.close();</script>");
		try{
			upFile.getFiles().getFile(0).saveAs("/music/"+fileName);			//保存文件到服务器
		}catch(Exception e){
			System.out.println("上传文件出现错误："+e.getMessage());
		}
		}else{
			out.println("<script>alert('该文件格式不符合要求，不能完成上传！');history.back(-1);</script>");
		}
}
%>
```

（2）添加歌曲信息完成后，还需要将该歌曲信息保存到数据库中。在表单提交后，将访问 URL 地址 song.do?action=add。通过该 URL 地址可以知道，在歌曲信息相关的 Action 实现类中，添加歌曲信息所涉及的方法为 adm_add()。在该方法中，首先获取歌曲的基本信息并进行转码，防止 SQL 注入；然后调用 SongDAO 类中的 insert()方法将歌曲信息保存到数据库；再根据返回结果设置相应的提示信息；最后将页面跳转到添加歌曲完成页面，显示相应的提示信息。adm_add()方法的具体代码如下：

```
public ActionForward adm_add(ActionMapping mapping, ActionForm form,
HttpServletRequest request, HttpServletResponse response) {
		SongForm songForm = (SongForm) form;
		songForm.setSongName(su.StringtoSql(songForm.getSongName()));			//歌曲名称
		songForm.setSinger(su.StringtoSql(songForm.getSinger()));				//演唱者
		songForm.setSpecialName(su.StringtoSql(songForm.getSpecialName()));	//专辑名
		int rtn=songDAO.insert(songForm);								//保存歌曲信息到数据库
		if(rtn>0){
			request.setAttribute("info", "歌曲添加成功！");
		}else{
			request.setAttribute("error","歌曲添加失败！");
		}
		return mapping.findForward("addok");								//将页面跳转到添加歌曲完成页面
}
```

在上面的代码中，调用了 SongDAO 类中的 insert()方法将歌曲信息保存到数据库。insert()方法比较简单，只需要执行将歌曲信息添加到数据库中的 SQL 语句即可，所以此处只给出向歌曲信息表中添加歌曲信息的 SQL 语句，详细代码参见源程序。

```
sql = "INSERT INTO tb_song (songName,singer,specialname,fileSize,fileURL,format,songType) VALUES('"+sf.getSongName()+"','"+sf.getSinger()+
"','"+sf.getSpecialName()+"','"+sf.getFileSize()+"','"+sf.getFileURL()+"','"+sf.getFormat()+"',"+sf.getSongTypeId()+")";
```

秘笈心法

心法领悟 590：带有上传表单组件的表单设置。

带有上传表单组件的表单设置与普通的表单设置是不同的，需要对<form>表单进行 enctype="multipart/form-data"属性设置。

实例 591 以顺序和随机方式进行歌曲连播

光盘位置：光盘\MR\23\591

中级

实用指数：★★★☆

本实例中提供了顺序和随机方式来播放歌曲。例如，在新歌速递区中，选中“有多少爱可以重来”、“红遍全球”和“改变自己”前面的复选框后，单击“歌曲连播”超链接，将打开歌曲连播窗口。在该窗口中，可以选择顺序播放或随机播放，如图 23.26 所示。默认情况下采用的是顺序播放。



图 23.26　以顺序或随机方式播放歌曲

关键技术

通过 HTML 提供的<object>标签可以调用指定媒体播放器播放多媒体文件。<object>标签的语法格式如下：

```
<object classid="clsid" id="id" width="width" height="height">
</object>
```

其中，classid 属性用于指定使用的浏览器插件（例如，调用 Windows Media Player 播放器，可以将 classid 属性的值设置为 clsid:6BF52A52-394A-11D3-B153-00C04F79FAA6 或 clsid:22D6F312-B0F6-11D0-94AB-0080C74C7E95；调用 RealPlayer 播放器，可以将 classid 属性的值设置为 clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA）；id 属性用于指定该对象的 ID 值；width 属性用于指定媒体播放器的宽度；height 属性用于指定媒体播放器的高度。

在<object>标签中，还可以包括<param>子标签，该标签用于设置<object>标签所调用的媒体播放器的相关属性。下面将对比较常用的属性进行介绍。

- url：用于指定要播放文件的路径。可以是绝对路径，也可以是相对路径。
- volume：用于控制音量，值为 0～100 之间的整数，表示 0%～100%。
- playcount：用于指定播放次数。
- enableerrordialogs：用于指定是否启用错误提示报告。
- autostart：用于指定是否自动播放，1 表示自动播放，0 表示不自动播放。

（1）在歌曲信息相关的 Action 实现类中，编写实现歌曲连播的方法 continuePlay()。在该方法中，首先获取要进行连播歌曲的 ID，并将获取的 ID 数组连接为一个以逗号分隔的字符串；然后获取歌曲文件的路径；再调用 SongDAO 类中的 continuePlay()方法获取要播放歌曲的信息，并保存到 HttpServletRequest 对象中；最后将页面重定向到歌曲连播页面。continuePlay()方法的具体代码如下：

```
public ActionForward continuePlay(ActionMapping mapping, ActionForm form,
            HttpServletRequest request, HttpServletResponse response) {
        SongForm songForm = (SongForm) form;
        String playID = "";
        //将要播放的歌曲 ID 连接为一个以逗号分隔的字符串
        for (int i = 0; i < songForm.getPlayId().length; i++) {
            playID = playID + songForm.getPlayId()[i] + ",";
        }
        playID = playID.substring(0, playID.length() - 1);                //去除尾部的逗号
        String realPath = request.getRealPath("/");
        String url = request.getRequestURL().toString();
        url = url.substring(0, url.lastIndexOf("/") + 1) + "music/";
        request.setAttribute("songNameList", songDAO.continuePlay(playID, url,
                "ORDER BY upTime DESC"));                                   //获取连续播放的歌曲
        return mapping.findForward("continuePlay");
}
```

在上面的方法中调用了 SongDAO 类中的 continuePlay()方法，在该方法中，将根据传递的歌曲 ID 查询要播放的歌曲，并保存到 List 集合中。查询要播放歌曲的 SQL 语句如下：

```
SELECT * FROM tb_song WHERE id IN ("+playId+") "+condition
```

（2）编写歌曲连播页面，实现按顺序或随机方式播放歌曲。

① 在页面中添加一个表单及一张 3 行 2 列的表格，并将该表格第 1 行的两个单元格合并为一个单元格，将

其 id 属性设置为 myPlayer，用于显示播放器。关键代码如下：

```
<form name="form1" method="post" action="">
<table width="363" height="185" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td colspan="2" id="myPlayer">正在加载播放器……
    </td>
    </tr>
      ……          //此处省略了其他行的代码
</table>
</form>
```

② 在步骤①创建的表格的第 2 行左侧单元格中输入提示性文字“播放列表”，在右侧的单元格中添加一个名为 playType 的下拉列表框，其中包括“顺序播放”和“随机播放”两个选项。具体代码如下：

```
  <tr>
    <td width="60" height="35">播放列表</td>
    <td width="303" align="right"><select name="playType" id="playType">
      <option value="0" selected>顺序播放</option>
      <option value="1">随机播放</option>
    </select></td>
</tr>
```

③ 将步骤①创建的表格的第 3 行合并为一个单元格，并在该单元格中添加一个用于显示播放列表的列表框。具体代码如下：

```
  <tr>
    <td colspan="2"><select name="playList" size="10" id="playList" ondblclick="list_dblClick();" style=" width:360px">
      <logic:iterate id="song" name="songNameList" type="com.model.SongForm" scope="request" indexId="ind">
        <option value="<bean:write name="song" property="fileURL" filter="true"/>">
        <bean:write name="song" property="songName" filter="true"/>
      </logic:iterate>
    </select></td>
  </tr>
```

④ 编写自定义的 JavaScript 函数 init()，用于在页面加载后调用相应的播放器，按顺序方式播放歌曲列表。在该函数中将首先判断客户端是否安装了 Windows Media Player 播放器，如果已安装，将动态加载该播放器，进行歌曲连播，否则判断客户端是否安装了 RealPlayer 播放器，如果已安装，将加载该播放器进行歌曲连播，否则将提示安装相关播放器。init()函数的具体代码如下：

```
function init(){
        //检测是否安装了 Windows Media Player 播放器
        if(isInsalled('{22d6f312-b0f6-11d0-94ab-0080c74c7e95}')){
                document.getElementById("myPlayer").innerHTML="<object classid='clsid:22D6F312-B0F6-11D0-94AB-0080C74C7E95'
id='wghMediaPlayer'name='wghMediaPlayer' width='360' height='64'><param name='volume' value='100'><param name='playcount'
value='100'><param name='enableerrordialogs' value='0'><param name='ShowStatusBar' value='-1'></object> ";
                document.getElementById("wghMediaPlayer").AutoRewind=false;
                document.getElementById("wghMediaPlayer").AutoStart=true;               //设置自动播放
                document.getElementById("wghMediaPlayer").SendPlayStateChangeEvents=true;
        document.getElementById("wghMediaPlayer").attachEvent("PlayStateChange",checkPlayStatus);
                if(form1.playList.options.length>0){
                        form1.playList.options[0].selected=true;
                        document.getElementById("wghMediaPlayer").fileName=form1.playList.value;
                        document.getElementById("wghMediaPlayer").play();
                }
        }else if(checkRealPlayer){                                               //检测是否安装了 RealPlayer 播放器
                        document.getElementById("myPlayer").innerHTML="<object classid='clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA'
id='wghMediaPlayer' name='wghMediaPlayer' width='360' height='64'><param name='volume' value='100'><param name='playcount'
value='100'><param name='enableerrordialogs' value='0'><param name='ShowStatusBar' value='-1'><param name='SendPlayStateChangeEvents'
value='1'></object> ";
                document.getElementById("wghMediaPlayer").AutoRewind=false;
                document.getElementById("wghMediaPlayer").AutoStart=true;
                if(form1.playList.options.length>0){
                        realPlayerPlay();                                       //连续播放
                }
        }else{
                alert("请安装 Windows Media Player 或 RealPlayer 播放器！");
                window.close();
        }
}
```

⑤ 编写自定义的 JavaScript 函数 checkPlayStatus()，用于当使用 Windows Media Player 播放器时，在播放状态改变时连续播放歌曲。具体代码如下：

```
function checkPlayStatus(){
	try{
		if(document.getElementById("wghMediaPlayer").PlayState==0){
	document.getElementById("wghMediaPlayer").detachEvent("PlayStateChange",checkPlayStatus);
			document.getElementById("wghMediaPlayer").stop();			//停止播放
		if(form1.playType.value==0){						//表示顺序播放
			if(form1.playList.options.selectedIndex<form1.playList.options.length-1){
				form1.playList.options[form1.playList.options.selectedIndex+1].selected=true;
			}else{
				form1.playList.options[0].selected=true;			//设置第一个列表项被选中
			}
		}else{										//随机播放
			//生成一个 0 到歌曲总数-1 的随机整数
			var randomValue=Math.round(Math.random() * (form1.playList.options.length - 1)) ;
			form1.playList.options[randomValue].selected=true;
	}
			document.getElementById("wghMediaPlayer").fileName=form1.playList.value;
			document.getElementById("wghMediaPlayer").play();			//开始播放
setTimeout('document.getElementById("wghMediaPlayer").play();document.getElementById("wghMediaPlayer").attachEvent("PlayStateChange",checkPlayStatus);',1000);
		}
	}catch(e){
		alert("出错了");
	}
}
```

⑥ 编写自定义的 JavaScript 函数，用于当使用 RealPlayer 播放器时，连续播放歌曲。具体代码如下：

```
function realPlayerPlay(){
	if(document.getElementById('wghMediaPlayer').getPlayState()==0){
		document.getElementById("wghMediaPlayer").doStop();			//停止播放
		if(form1.playType.value==0){						//表示顺序播放
			if(form1.playList.options.selectedIndex<form1.playList.options.length-1){
				form1.playList.options[form1.playList.options.selectedIndex+1].selected=true;
			}else{
				form1.playList.options[0].selected=true;			//设置第一个列表项被选中
			}
		}else{										//随机播放
			//生成一个 0 到歌曲总数-1 的随机整数
			var random Value=Math.round(Math.random() * (form1.playList.options.length - 1)) ;
			form1.playList.options[randomValue].selected=true;
		}
		document.getElementById("wghMediaPlayer").Source=form1.playList.value;
		document.getElementById("wghMediaPlayer").doPlay();			//开始播放
	}
	var timer=setTimeout("realPlayerPlay()",1000);
}
```

⑦ 在进行歌曲连播时，为了让用户可以选择指定的歌曲开始播放，还需要添加双击列表框中指定歌曲时开始播放的功能。实现该功能时，需要编写一个自定义的 JavaScript 函数，这里为 list_dblClick()。在该函数中，也需要根据选择的播放器从指定的歌曲开始播放。list_dblClick()函数的具体代码如下：

```
function list_dblClick(){
	if(isInsalled('{22d6f312-b0f6-11d0-94ab-0080c74c7e95}')){
	document.getElementById("wghMediaPlayer").detachEvent("PlayStateChange",checkPlayStatus);
		//将列表框的值指定给 Media Player 播放器
		document.getElementById("wghMediaPlayer").fileName=form1.playList.value;
		document.getElementById("wghMediaPlayer").play();		//开始播放
	setTimeout('document.getElementById("wghMediaPlayer").play();document.getElementById("wghMediaPlayer").attachEvent("PlayStateChange",
checkPlayStatus);',1000);
	}else if(checkRealPlayer){						//检测是否安装 RealPlayer 播放器
		//将列表框的值指定给 RealPlayer 播放器
		document.getElementById("wghMediaPlayer").Source=form1.playList.value;
		document.getElementById("wghMediaPlayer").doPlay();		//开始播放
	}
}
```

秘笈心法

心法领悟 591：通过 JavaScript 函数实现复选框的全选。

本实例中定义了多个复选框，用户可以通过“全选”超链接设置复选框的全选。在此主要是通过 JavaScript 函数实现复选框的全选和反选，代码如下：

```
function CheckAll(elementsA,elementsB){
        for(i=0;i<elementsA.length;i++){
                elementsA[i].checked = true;
        }
        if(elementsB.checked ==false){
                for(j=0;j<elementsA.length;j++){
                        elementsA[j].checked = false;
                }
        }
}
```

23.6 校内数码相册



实例 592　以幻灯片方式播放数码相片　中级

光盘位置：光盘\MR\23\592　实用指数：★★☆

实例说明

所谓以幻灯片方式播放数码相片，是指页面中的相片交替显示，并在两张相片切换时插入变换效果。在本实例中，页面中的相片将按顺序播放，并且在两张相片切换时采用水平棋盘式变换效果，如图 23.27 所示。单击▶按钮可显示下一张相片，单击◀按钮会显示上一张相片，如果不单击任何按钮则会自动循环播放。



图 23.27　以幻灯片方式播放数码相片

关键技术

获取 request 范围内的 address 对象，并强制转换成 String 类型的数组 address，该对象数组存放的是浏览相片的地址，之后通过 for 循环将 address 数组中的内容通过逗号隔开，形成一个 String 类型的字符串，生成的对象是 newAddress。

（1）创建 PhotoServlet 类，在该类中编写查询相册信息的方法，将查询出的所有相片地址保存在数组中，并将请求转发到相片浏览页面。代码如下：

```
//幻灯片浏览
public void queryPhotoSlide(HttpServletRequest request,
                HttpServletResponse response) throws ServletException, IOException {
        data = new OperationData();
        UserInfo userInfo = (UserInfo) request.getSession().getAttribute(
                        "userInfo");                                    //获取客户端 Session 中的指定对象
        String username = userInfo.getUsername();                      //获取用户名
        String type = com.wy.tools.Encrypt.toChinese(request
                        .getParameter("type"));                         //获取网站类型名称
        String condition = "username ='" + username + "' and photoType = '"
                        + type + "'";                                   //设置查询的条件：以用户名与相册名称为条件
        List list = data.photo_queryList(condition);                   //执行查询操作
        String address[] = new String[list.size()];                    //设置相片存储位置的数组内容
        for (int i = 0; i < list.size(); i++) {
```

```
                Photo photo = (Photo) list.get(i);
                address[i] = photo.getPhotoAddress();                   //对查询结果中的相片存放位置一一进行赋值
        }
        request.setAttribute("address", address);                       //将相片地址数组存放在 request 范围内
        request.getRequestDispatcher("photoShowSlide.jsp").forward(request,
                        response);
}
```

（2）创建 photoShowSlide.jsp 页，从请求中读取出相片的地址。代码如下：

```
<%
String newAddress="";
String[] address=(String[])request.getAttribute("address");
for(int i=0;i<address.length;i++){
      newAddress+="'"+address[i]+"',";
}
newAddress=newAddress.substring(0,newAddress.length()-1);
%>
```

（3）定义一个一维数组，用于存放要播放的相片路径。代码如下：

```
<SCRIPT language=javascript type=text/javascript>
var sImgArr=new Array(<%=newAddress%>);
</SCRIPT>
```

（4）编写以幻灯片方式播放相片的自定义函数。代码如下：

```
<SCRIPT language=javascript type=text/javascript>
function SlideImg(index){
      gIndex = index;
      if ('Microsoft Internet Explorer' == navigator.appName)    {
            document.images["slideImg"].filters.item(0).Apply();
      }
      document.images["slideImg"].src = sImgArr[index];
      if ('Microsoft Internet Explorer' == navigator.appName){
            document.images["slideImg"].filters.item(0).play();
      }
}
</SCRIPT>
```

（5）编写单击“上一张”或者“下一张”按钮时调用的自定义函数。代码如下：

```
function NextImg(){                                     //显示下一张相片
      gIndex = ((gIndex+1)>=sImgArr.length?0:(gIndex+1));
      SlideImg(gIndex);
}
function PrevImg(){                                     //显示上一张相片
      gIndex = ((gIndex-1)<0?(sImgArr.length-1):(gIndex-1));
      SlideImg(gIndex);
}
```

（6）在页面中适当位置加入相片，并设置其滤镜效果。代码如下：

```
<img src="images/1.jpg" name="slideImg" width="300" height="231" id="slideImg" style="BORDER-TOP: #000 1px solid; FILTER: revealtrans
(duration=2.0,transition=10); BORDER-BOTTOM: #000 1px solid">
```

秘笈心法

心法领悟 592：幻灯片自动播放。

实现幻灯片自动播放，可以通过 JavaScript 的 setInterval()函数实现。代码如下：

```
var sid;
function inislide(){                          //设置自动运行
      if(sid==null){
            sid = setInterval('NextImg()', 3000);
      }
}
```

实例 593	创建相册分类并上传相片 光盘位置：光盘\MR\23\593	中级 实用指数：★★★

实例说明

无论是什么样的相册系统，相片上传都是必备的一项功能，这是获取相片的唯一来源。在本实例中，当用

户登录后，将进入分栏显示相片类别页面。单击“请您上传自己的相片”超链接，将进入相片上传页面，如图 23.28 所示。

图 23.28　相片上传页面

关键技术

实现照片的上传功能，本实例应用了 jspSmartUpload 组件。该组件用于实现文件的上传，并且操作非常方便，在此就不详细说明，具体的使用方法可参见相应的 API。

（1）在相片上传页面中，用户在相应表单中输入完整的信息后，单击“上传”按钮，网页将转向一个 URL photoServlet?info=userUploadPhoto。从该 URL 地址中可以知道相片上传涉及 PhotoServlet 实现类，该类中的 userUploadPhoto()方法用于实现相片的上传。userUploadPhoto()方法代码如下：

```
public void user_uploadPhoto(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {
    data = new OperationData();
    com.jspsmart.upload.SmartUpload su = new com.jspsmart.upload.SmartUpload();
    String information = "您输入的数据有误，添加相片失败！";
    try {
     su.initialize(this.getServletConfig(), request, response);                    //设置上传操作的初始化
     su.setMaxFileSize(2 * 1024 * 1024);                                           //设置上传文件的大小
     su.upload();
     Files files = su.getFiles();                                                  //获取上传的文件
          for (int i = 0; i < files.getCount(); i++) {
           File singleFile = files.getFile(i);                                     //获取上传文件中的单个文件
           String fileType = singleFile.getFileExt();                              //获取上传文件的扩展名
           String[] type = { "JPG", "jpg", "gif", "bmp", "BMP" };                  //设置上传文件的扩展名
           int place = java.util.Arrays.binarySearch(type, fileType);              //判断上传文件的类型是否正确
           String code = su.getRequest().getParameter("code");                     //获取表单中验证码内容
           String codeSession = (String) request.getSession().getAttribute("rand");
                                                                                   //获取客户端 Session 中验证码的值
           if (code.equals(codeSession)) {                                         //判断验证码是否正确
               if (place != -1) {                                                  //判断文件扩展名是否正确
     if (!singleFile.isMissing()) {                                                //判断该文件是否被选择
           String photoName = su.getRequest().getParameter("photoName")+ i;        //获取相片名称
           String photoType = su.getRequest().getParameter("photoType");           //获取相册名称
           String photoTime = su.getRequest().getParameter("photoTime");           //获取相册上传时间
           String username = su.getRequest().getParameter("username");             //获取上传用户名
           String photoSize = String.valueOf(singleFile.getSize());                //获取上传文件大小
           String filedir = "savefile/"+ System.currentTimeMillis() + "."+ singleFile.getFileExt();
           //以系统时间作为上传文件名称，设置上传文件的完整路径
           String smalldir = "saveSmall/"+ System.currentTimeMillis() + "."+ singleFile.getFileExt();
           Photo photo = new Photo();
               /****************将 photo 对象中的属性进行逐个赋值*******************/
           photo.setPhotoName(photoName);
           photo.setPhotoType(photoType);
           photo.setPhotoSize(photoSize);
           photo.setPhotoTime(photoTime);
           photo.setUsername(username);
           photo.setPhotoAddress(filedir);
           photo.setSmallPhoto(smalldir);
           /************************************************************************/
     if (data.photo_save(photo)) {                                                 //实现上传操作的 SQL 语句
           singleFile.saveAs(filedir, File.SAVEAS_VIRTUAL);                        //执行上传操作
           com.wy.tools.ImageUtils.createSmallPhoto(request.getRealPath("/" + filedir),
                  request.getRealPath("/" + smalldir));
           information = "您添加相片成功!";
```

```
                                                }
                                    }
                              }
                        }
                  }
            } catch (Exception e) {
                  System.out.println(e);
            }
            request.setAttribute("information", information);
            request.getRequestDispatcher("user_upLoadPhoto.jsp").forward(request,response);
      }
```

（2）在 uploadPhoto()方法中，实现相片添加是通过调用 Dao 类中的 photo_save()方法实现的。最后将执行的结果通过 return 关键字返回。photo_save()方法代码如下：

```
public boolean photo_save(Photo photo) {
            connection = new JDBConnection();                            //将 JDBConnection 对象进行实例化
            sql = "insert into tb_photo values ('" + photo.getPhotoName() + "','"
                  + photo.getPhotoSize() + "','" + photo.getPhotoType() + "','"
                  + photo.getPhotoTime() + "','" + photo.getPhotoAddress()
                  + "','" + photo.getUsername() + "',0,'" + photo.getSmallPhoto()
                  + "')";                                                //设置保存相片数据的 SQL 语句
            boolean flag = connection.executeUpdate(sql);               //执行保存相片信息操作的 SQL 语句
            connection.closeConnection();                                //关闭数据库连接
            return flag;                                                 //将 flag 对象作为这个方法的返回结果
      }
```

秘笈心法

心法领悟 593：批量上传相片。

如图 23.28 所示，在“相片位置”栏中，用户可以单击“增加”按钮增加“相片位置”文本框；如果文本框增加过多，可以单击“移除”按钮对文本框进行移除。其中对表单的增加或移除操作涉及的 JavaScript 语言的代码如下：

```
<script type="text/javascript">
      function addMore(){
      var td = document.getElementById("more");
      var br= document.createElement("br");
      var input= document.createElement("input");
      var button= document.createElement("input");
      input.type = "file";
      input.name = "file";
      button.type = "button";
      button.value = "移除...";
      button.onclick = function(){
         td.removeChild(br);
         td.removeChild(input);
         td.removeChild(button);
      }
      td.appendChild(br);
      td.appendChild(input);
      td.appendChild(button);
      }
</script>
```

通过上面的 JavaScript 脚本语言的设置，可以对文件位置的上传表单进行动态的增加或删除操作，这样有助于实现批量上传相片。

实例 594	浏览和管理上传相片 光盘位置：光盘\MR\23\594	中级 实用指数：★★★☆

实例说明

在本实例中，无论是已登录用户还是未登录用户都可以实现对相片的详细查询。在滚动浏览相片页面中，

单击放大的相片，可以对相片的大小、上传时间和相片的拥有者等信息进行详细的查询和删除操作，如图 23.29 所示。



图 23.29　浏览和管理上传相片

关键技术

在滚动浏览相片页面中，单击放大的相片超链接的代码如下：

```
<a href="photoServlet?info=queryOnePhoto&id=<%=photo.getId()%>" id="toward">
        <img src="<%=photo.getPhotoAddress()%>" width="400" height="300" id="images11">
</a>
```

在上述代码中，超链接 id 参数为查询当前相片的 id 值，根据这个 id 值可以查询相片在数据库中的全部信息。

单击放大的相片超链接后，网页将转向一个 URL“photoServlet?info= queryOnePhoto”。从该 URL 地址中可以知道相片详细查询涉及 PhotoServlet 实现类，该类中的 queryOnePhoto()方法用于实现相片的详细查询。

##

（1）在 PhotoServlet 中编写 queryOnePhoto()方法，代码如下：

```
public void queryOnePhoto(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
          data = new OperationData();
          Integer id = Integer.valueOf(request.getParameter("id"));          //获取页面中相册的 id 号
          String condition = "id = '" + id + "'";                            //设置以 id 号为查询条件
          List list = data.photo_queryList(condition);                       //执行查询的方法
          Photo photo = null;
          if (list.size() == 1) {                                            //由于 id 号的值在数据库中是唯一的，因此只存在一组数据
        photo = (Photo) list.get(0);
          }
          request.setAttribute("photo", photo);                              //将查询的结果保存在 request 范围内
          try {
        request.getRequestDispatcher("photoShow.jsp").forward(request,response);
        return;
          } catch (Exception e) {

          }
      }
```

（2）创建相片详细查询页面 photoShow.jsp，首先获取 request 范围内的 photo 对象，该对象中存放着相片

的详细信息，并将结果强制转型后赋值给 photo 对象，之后通过 photo 对象的 getXXX()方法将相片信息显示在页面中。代码如下：

```
<%
Photo photo=(Photo)request.getAttribute("photo");
%>
<table width="753" height="499" border="0">
    <tr>
      <td width="438" valign="top">
      <img src="<%=photo.getPhotoAddress()%>" height="315" width="403" id="images11"></td>
      <tr>
          <td width="75" height="24" align="right">相片名称：</td>          //显示相片名称
          <td width="120"><%=photo.getPhotoName()%></td>
        </tr>
        <tr>
          <td height="24" align="right">相册名称：</td>                    //显示相册名称
          <td><%=photo.getPhotoType()%></td>
        </tr>
        <tr>
          <td height="24" align="right">相片大小：</td>                    //显示相片大小
          <td><%
          float size = Integer.valueOf(photo.getPhotoSize()) /1024/;
          out.print(size);
          %>KB</td>
        </tr>
        <tr>
          <td height="24" align="right">上传时间：</td>                    //显示上传时间
          <td><%=photo.getPhotoTime()%></td>
        </tr>
        <tr>
          <td height="24" align="right">上传用户：</td>                    //显示上传用户
          <td><%=photo.getUsername()%></td>
        </tr>
</table>
```

（3）在 PhotoServlet 类中编写 user_deletePhoto()方法，用于实现相片的删除操作。代码如下：

```
public void user_deletePhoto(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html;charset=GBK");
    PrintWriter out = response.getWriter();
    data = new OperationData();
    Integer id = Integer.valueOf(request.getParameter("id"));       //从页面中获取要删除相片的 id 号
    String condition = "id=" + id;                                   //设置以 id 为查询条件
    List list = data.photo_queryList(condition);                     //根据 id 值查询相片的一组信息
    String address = null;                                           //设置存放服务器端地址对象
    String print = null;
    String type = null;                                              //设置相片所在相册对象
    if (list.size() == 1) {                                          //判断查询的集合内容是否存在一组数据
        Photo photo = (Photo) list.get(0);
        address = photo.getPhotoAddress();                           //获取数据库中相片存放服务器端的地址
        print = photo.getPrintAddress();
        type = photo.getPhotoType();                                 //获取相片所在相册的类型
    }
    String path = request.getRealPath("/" + address);               //获取文件的实际地址
    data.photo_delete(id);                                           //删除相片所对应的 SQL 语句
    //下面的操作是根据文件的所在位置进行删除操作
    java.io.File file1 = new java.io.File(path);
    if (file1.exists()) {
        file1.delete();
    }
    String printPath = request.getRealPath("/" + print);
    java.io.File file2 = new java.io.File(printPath);
    if (file2.exists()) {
        file2.delete();
    }
    //将文件的类型保存在 request 范围内
```

```
            request.setAttribute("type", type);
            request.getRequestDispatcher("dealwith.jsp").forward(request, response);
        }
```

（4）删除相片存放在数据表的信息使用的是 OperationData 类的方法 photo_delete()。在 photo_delete()方法中，根据参数 id 值设置删除相片的 SQL 语句并执行该 SQL 语句，之后将执行的结果通过 return 关键字返回。photo_delete()方法的具体代码如下：

```
//相片的删除操作，该方法以相片的 id 为条件
public boolean photo_delete(Integer id) {
   connection = new JDBConnection();                            //实例化 JDBConnection 类的对象
   sql = "delete from tb_photo where id=" + id + "";            //设置删除相片的 SQL 语句
   boolean flag = connection.executeUpdate(sql);                //执行删除的 SQL 语句，并将执行结果赋值给 flag 对象
   connection.closeConnection();                                //关闭数据库连接
   return flag;                                                 //将 SQL 语句的执行结果作为这个方法的返回值
}
```

秘笈心法

心法领悟 594：为相片添加水印效果。

为相片添加水印就是在相片中加入指定文字（一般是网站的名称和地址），其作用是最大限度地防止盗用，同时也能起到标识相片的功能。本实例中，添加水印效果需要用到 JavaBean 的 createMark()方法，主要就是应用 java.awt 包中的相关绘图工具类实现在相片中添加文字效果。具体代码如下：

```
public static boolean createMark(String filePath, String printPath,String markContent) {
        ImageIcon imgIcon = new ImageIcon(filePath);                  //读取源相片内容
        Image theImg = imgIcon.getImage();                            //获取相片的信息
        int width = theImg.getWidth(null);                            //获取相片的宽度
        int height = theImg.getHeight(null);                          //获取相片的长度
        BufferedImage bimage = new BufferedImage(width,height,
                        BufferedImage.TYPE_INT_RGB);
        Graphics2D g = bimage.createGraphics();
        g.setColor(Color.red);                                        //设置文字颜色为红色
        g.drawImage(theImg, 0, 0, null);
        Font font = new Font(markContent, Font.BOLD, 200);            //对文字进行加粗
        g.setFont(font);
        g.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 0.5f));//50%透明
        g.rotate(0.3f);                                               //文字的旋转角度
        g.drawString(markContent, width/3, height/3);                 //绘制水印的位置
        g.dispose();
        try {                                                         //通过输出流生成相片内容
         FileOutputStream out = new FileOutputStream(printPath);
        JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(out);
        JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bimage);
        param.setQuality(100f, true);
        encoder.encode(bimage, param);
        out.close();
        } catch (Exception e) {
          e.printStackTrace();
        return false;
        }
        return true;
      }
```



实例 595　数码相册分类管理　　中级

光盘位置：光盘\MR\23\595　　实用指数：★★★

实例说明

在网站首页中，单击“我的相册”超链接，根据 URL 地址将页面转发到 user_queryPhoto.jsp 页面，在其中分栏显示出当前用户上传相片的类别信息与该类别信息相对应的相片，运行结果如图 23.30 所示。



图 23.30　分栏显示相片类别

关键技术

当用户成功登录后，单击网站首页导航区域中的“我的相册”超链接，可以对当前用户上传的相片类别进行查询。当用户未登录时，单击导航区域中的“我的相册”超链接，则通过 JavaScript 脚本语言弹出提示用户未登录的对话框。根据“我的相册”超链接判断用户是否登录的代码如下：

```
<script language="javascript" src="js/js.js" type="text/javascript"></script>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:if test="${sessionScope.userInfo==null}">
<a href="#" onclick="javascript:checkUserInfo()" title="请您先登录">我的相册</a>
</c:if>
<c:if test="${sessionScope.userInfo!=null}">
<a href="photoServlet?info=userQueryPhoto" class="a1">我的相册</a>
</c:if>
```

在上述代码中，用户登录后，系统将用户登录信息 userInfo 对象保存在客户端的 Session 中，通过判断 Session 中是否存在指定对象，即可判断该用户是否登录。

（1）当用户成功登录后，单击“我的相册”超链接，将执行 photoServlet?info=userQueryPhoto。根据 web.xml 文件的配置信息可以知道，该操作执行的是 PhotoServlet 中的 userQueryPhoto()方法。该方法的代码如下：

```
//当用户成功登录后，实现登录用户查询相册的功能
public void user_queryPhoto(HttpServletRequest request,HttpServletResponse response) throws ServletException, IOException {
        data = new OperationData();
        UserInfo userInfo = (UserInfo) request.getSession().getAttribute(      "userInfo");
                                                                              //获取客户端存放在 Session 对象中的数据
        String username = userInfo.getUsername();                             //获取用户名
        String[] type = data.queryPhotoType(username);                        //根据用户名查询该用户上传相册的名称
        request.setAttribute("type", type);                                   //将相册类型存放在 request 范围内
        String condition = "username = '" + username + "'";
        List list = data.photo_queryList(condition);                          //根据用户名查询相册内容
        request.setAttribute("list", list);                                   //将查询的结果保存在 request 请求范围内
        request.getRequestDispatcher("user_queryPhoto.jsp").forward(request,response);
    }
```

（2）根据用户名查询该用户上传相册的操作调用的是 OperationData 类中的 queryPhotoType()方法，该方法的代码如下：

```
    //以用户名为条件，查询该用户上传相册的名称
    public String[] queryPhotoType(String username) {
     String[] type = null;                                       //设置 type 数组，该数组保存用户上传相册的名称
     sql = "select photoType from tb_photo   where username='" + username
                 + "' group by photoType";                       //设置分组查询的 SQL 语句
     connection = new JDBConnection();                           //将 JDBConnection 对象进行实例化
     ResultSet rs = connection.executeQuery(sql);
                                                                 //执行查询 SQL 语句，并将查询结果保存在 rs 对象中
     try {
            rs.last();                                           // rs 指针指向最后一组数据
            int length = rs.getRow();                            //查询当前记录数
            type = new String[length];                           //将数据的长度进行设置
            rs.beforeFirst();                                    //将 rs 指针指向最前面的数据库
```

```
            int i = 0;                                    //设置 i 变量，用来记录循环的次数
        // rs 对象进行循环
            while (rs.next()) {
            type[i++] = rs.getString("photoType");        //将数据中的每个对象进行赋值
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            connection.closeConnection();
        }
        return type;                                      //将查询的结果数组进行返回
    }
```

（3）在 user_queryPhoto.jsp 页面中，首先判断 request 范围内的 type 对象是否为空，如果为空，则说明当前用户并没有上传任何相片；如果不为空，则将获取的内容强制转型为 String 类型的数组，循环该数组中的内容并显示在页面中。其中在显示相片类别的过程中，获取 request 范围内的 list 对象，之后循环 List 集合中的内容。如果 String 类型的数组中的相片类别名称与 list 对象中的相片类别名称一致，则在页面中显示相片的内容。user_queryPhoto.jsp 页面的代码如下：

```
<c:if test="${empty type}">
        <c:out value="您还没有上传自己的相片"/>
</c:if>
<c:if test="${!empty type}">
    <table width="139" border="0" cellpadding="0" cellspacing="0">
    <%
      String[] typePhoto=(String[])request.getAttribute("type");
      int lineCount=5;                                   //设置一行显示 5 个类别名称
      int typeLength=typePhoto.length;                   //获取当前用户上传相片类别的数量
      int rowCount=typeLength/lineCount;                 //计算有多少行数
      if(typeLength%lineCount!=0){
        rowCount++;
      }
      List list=(List)request.getAttribute("list");      //获取 request 范围内的 list 对象
      %>
      <tr>
     <%
      for(int i=0;i<typeLength;i++){                     //循环显示相片类型名称
     %>
     <td width="149" height="153" align="center">
       <table width="128" height="142" border="0">
  <tr>
    <td width="118" height="111" align="center">
    <%
        for(int j=0;j<list.size();j++){                  //循环相片 List 集合对象
            Photo photo=(Photo)list.get(j);
            if(photo.getPhotoType().equals(typePhoto[i])){  //判断相片类别名称是否一致
    %>
       <a href="photoServlet?info=queryPhotoList&type=<%=photo.getPhotoType()%>">
<img src="<%=photo.getSmallPhoto()%>">
        </a>
    <% break;}}%>
    </td>
  </tr>
  <tr>
    <td width="122" height="28"><%=typePhoto[i]%> </td>
  </tr>
</table>
</td>
<%
if(i%lineCount==lineCount-1){
    out.print("</tr><tr>");
            }
 }
if(rowCount*lineCount-typeLength>0){                     //以下是换行操作的实现过程
    int overCount=rowCount*lineCount-typeLength;
      for(int j=0;j<overCount;j++){
            out.print("<td align=center width=155> </td>");
```

```
    }
}
%>
</tr>
</table>
</c:if>
```

秘笈心法

心法领悟 595：<c:if>标签。

在判断 Session 中是否存在用户信息操作过程中应用到了<c:if>标签，该标签可以根据不同的条件处理不同的业务，即执行不同的程序代码与 Java 中的 if 语句的功能一样。

23.7　仿百度知道之明日知道

实例 596	在线提问 光盘位置：光盘\MR\23\596	中级 实用指数：★★★☆

实例说明

本实例实现的是明日知道论坛的在线提问功能，相当于论坛中发表文章的功能。运行程序，登录明日知道论坛后单击“我要提问”按钮，即可进入添加问题的页面，如图 23.31 所示。输入问题信息，单击“发表文章”按钮，所提出的问题就会显示在论坛的问题列表中。

图 23.31　在线提问

关键技术

整个程序是使用 SSH2 框架实现的，通过 Struts2 进行业务逻辑的控制；当单击“发表文章”按钮时，将调用 Action 类中的添加文章方法，在该方法中获取表单数据并封装在对象中；然后调用 DAO 层相关方法，使用 Hibernate 来持久化对象。在整个过程中，还将使用 Spring 来实现 Bean 的依赖注入关系和事务的管理。

（1）用户填写信息完毕，系统会校验填写内容的合法性，以防止不合法的数据对系统造成破坏，例如，非法字符、超长文字等。添加文章表单代码和校验代码如下：

```
<form action="articleAction_addArticle" method="post"
id="addArticleForm">
<table>
<tr>
<td class="huise">文章标题：</td>
```

```
<td><input typoe="text" name="article.title"
id="title"></td>
</tr>
<tr>
<td class="huise">所属类型：</td>
<td>
<select name="article.articleTypeName" id="type">
        <option value="">请选择</option>
                <option value="Visual Basic">Visual Basic</option>
                <option value="Visual C++">Visual C++</option>
                <option value="Java">Java</option>
                <option value="Java web">Java web</option>
                <option value="C#">C#</option>
                <option value="ASP.NET">ASP.NET</option>
                <option value="PHP">PHP</option>
                <option value="ASP">ASP</option>
                <option value="其他">其他</option>
        </select>
</td>
</tr>
<tr>
<td class="huise">文章内容：</td>
        <td><textarea name="article.content" cols="80"
rows="10"    id="content"></textarea></td>
</tr>
</table>
<p align="center"><input type="button" value="发表文章"
onclick="addArticle1()" /></p>
</form>
```

（2）页面将参数传给 Action，Action 将参数封装为文章对象传给 DAO 层，DAO 层调用保存方法将文章信息存入数据库。文章对象类代码如下：

```
public class Article {
private Integer articleId = null;                   //文章主键 id
private String title = null;                        //标题
private String content = null;                      //内容
private Date emitTime = null;                       //发表时间
private Date lastUpdateTime = null;                 //最后更新时间
private String articleTypeName = null;              //文章类型名称
private User user = null;                           //文章作者
private ArticleType articleType = null;             //文章类型
private Set<Reply> replies = null;                  //文章回复
private Set<Scan> scans = null;                     //文章浏览
……                                                  //get()和 set()方法省略
}
```

（3）文章添加 DAO 层代码调用如下：

```
/**
* 添加文章
*/
public void addArticle(Article article) {
ArticleType articleType = this.getArticleTypeByName(article.getArticleTypeName());
article.setArticleType(articleType);
this.getHibernateTemplate().save(article);
}
```

心法领悟 596：jQuery 框架。

jQuery 是一个简洁、快速、灵活的 JavaScript 脚本库，由 John Resig 于 2006 年创建，旨在帮助开发人员简化 JavaScript 代码。JavaScript 脚本库类似于 Java 的类库，将一些工具方法或对象方法封装在类库中，以方便用户使用。

实例 597　问题回复

光盘位置：光盘\MR\23\597

中级

实用指数：★★★

实例说明

用户浏览论坛中的问题之后，可以对其进行回复，但前提是用户必须已经登录系统，否则不能对文章进行回复。文章回复效果如图 23.32 所示。

图 23.32　问题回复

关键技术

回复问题也很简单，就是向回复表中插入一条记录。需要注意的是，在回复问题时，需要获取到当前要回复问题的 id 并保存到回复表中，因为在回复表中需要将文章列表中的 id 作为外键，这样在删除问题时，才能将回复表中的相关信息一并删除。

设计过程

（1）回复文章，后台操作就是向回复表里插入一条记录。回复实体类代码如下：

```
public class Reply {
	private Integer replyId = null;				//回复主键 id
	private Date replyTime = null;				//回复时间
	private String content = null;				//回复内容
	private User user = null;				//回复用户
	private Article article = null;				//回复的文章
	……							//get()和 set()方法省略
}
```

（2）回复文章持久层代码如下：

```
/**
* 添加回复
*/
public void addReply(Reply reply) {
this.save(reply);
}
```

心法领悟 597：设置 Bean 对象由 Spring 进行管理。

在 Struts2 的配置文件中，添加以下<constant>元素，这样，所有的 Bean 对象就会交给 Spring 容器来维护管理。

```
<constant name="struts.objectFactory" value="spring" />
```



实例 598 修改问题

光盘位置：光盘\MR\23\598

中级

实用指数：★★☆

实例说明

用户只能修改自己发表的问题。从问题列表中选择一篇文章进入后，系统会判断该文章的作者是不是当前用户，如果是，系统才会显示“修改”按钮，用户才有权限对文章进行修改，如图 23.33 所示。

修改文章

文章标题：	SSH指的是什么
所属类型：	Java web
文章内容：	请问，ssH具体是什么意思？有人知道吗
	修　改

图 23.33　修改问题

关键技术

首先在 Action 中会根据当前问题的 id 调用 DAO 层的方法查询出当前的文章对象；在修改问题页面中，再应用 Struts2 标签获取到对象的属性值作为修改问题的默认内容；最后当用户修改问题内容后，会提交给 Action 中的更新方法执行更新。

设计过程

（1）要修改文章，首先要根据文章 id 查询出文章，对文章需要修改的属性赋值，然后执行更新。修改文章 ActicleAction 代码如下：

```
/**
 * 修改文章
 * @return
 */
public String updateArticle() {
        Article article = articleDao.querySingleArticle(this.article.getArticleId().toString());
        article.setLastUpdateTime(new Date());
        article.setTitle(this.article.getTitle());
        article.setContent(this.article.getContent());
        this.articleDao.updateArticle(article);
        this.article = articleDao.querySingleArticle(this.article.getArticleId().toString());
        return "singleArticle";
}
```

（2）在 DAO 层修改文章的方法 updateArticle()的代码如下：

```
public void updateArticle(Article article) {
        this.update(article);
}
```

心法领悟 598：Struts2 实现分页。

Struts2 对模型驱动支持得很好，可以在页面上很方便地获取业务 Bean 中的属性；同时，其标签库也是非常

强大的。鉴于 Struts2 的这些优点，可以将分页也定义成一个可以重用的组件，这将为后续开发省去不少麻烦。具体的代码参见配书光盘中源程序中的 pageUti.jsp 和 PageUtil.java。

实例 599　关闭提出的问题

光盘位置：光盘\MR\23\599

中级

实用指数：★★★

实例说明

本实例实现的是关闭提出的问题，其实就是将自己发布的问题删除。当用户登录后，在如图 23.34 所示页面中单击“删除”超链接，即可删除自己提出的问题。

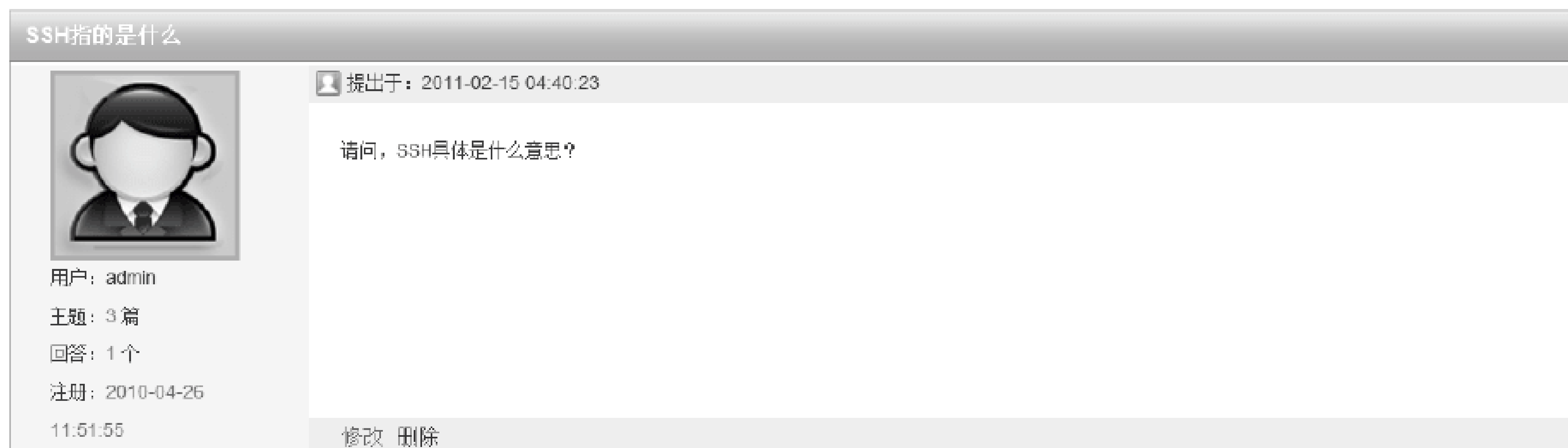


图 23.34　关闭提出的问题

关键技术

当单击“删除”超链接时，会将当前问题在数据库中的 id 传递给 Action，Action 调用 DAO 层的删除方法执行删除操作。

设计过程

（1）与修改文章一样，用户只能删除自己发表的文章。删除文章页面 singleArticle.jsp 的关键代码如下：

```
<s:a action="articleAction_deleteArticle" cssClass="hong">
<s:param name="article.articleId"
value="article.articleId"></s:param>
删除
</s:a>
```

（2）删除文章之后，再做一次查询，页面将跳转到“我的文章”列表。ActicleAction 的具体代码如下：

```
/**
 * 删除文章
 * @return
 */
public String deleteArticle() {
        articleDao.deleteArticle(this.article);                    //删除所选文章
        User user = new User();
        user.setUserId(this.getCurrUser().getUserId());            //设置用户信息
        this.article.setUser(user);
        //根据用户信息，查询其发表的所有文章
        this.myArticles = this.articleDao.queryAllArticleByUser(user, this.getFirstResult(), this.getMaxResults());
        return "myArticle";
}
```

心法领悟 599：删除该文章下的所有回复信息。

删除文章时，需要删除该文章下的所有回复信息以及浏览信息，否则数据库将会产生冗余数据。为了达到级联删除，只需在 Hibernate 映射文件中配置即可。Article.hbm.xml 的具体代码如下：

```
<set name="replies" inverse="true" cascade="all" order-by="replyTime desc">
<key column="articleId" />
<one-to-many class="Reply" />
</set>
<set name="scans" inverse="true" cascade="all" order-by="scanTime desc">
<key column="articleId" />
<one-to-many class="Scan" />
</set>
```

参数说明

❶ cascade 属性：cascade="all"为级联删除；order-by="replyTime desc"表示以时间倒序来排序。

❷ key 属性：<key column="articleId" />中的 articleId 为关联外键。

实例 600 搜索问题

光盘位置：光盘\MR\23\600

中级

实用指数：★★★

实例说明

本实例主要讲解关键字搜索。根据问题关键字查询，系统会用输入的关键字与所有文章的标题和内容进行任何位置的匹配，如果匹配成功则返回搜索结果，否则返回空。根据关键字搜索分为 3 种情况：在明日论坛不选择文章类型直接输入关键字进行搜索、在明日论坛首页选择一个文章类型再加上关键字进行搜索和进入论坛输入关键字进行搜索。如果选择文章类型，系统会在所有该类型下的文章中进行搜索；如果不选择文章类型，系统将在所有的文章中进行搜索。本实例运行结果如图 23.35 所示。



图 23.35　根据关键字搜索问题

明日知道模块的整体采用了 SSH2 的框架模式，由于 Spring 将 Hibernate 集成进来，并对 Hibernate 作了数据源和事务封装，因此不用单独编写额外代码管理 Hibernate 的事务处理，可以把主要精力放在企业级业务逻辑上。关键代码如下：

```
<!-- 配置 sessionFactory -->
<bean id="sessionFactory"class="org.springframework.orm.hibernate3.LocalSessionFactoryBea">
<property name="configLocation">
<value>classpath:hibernate.cfg.xml</value>
</property>
</bean>
<!-- 配置事务管理器 -->
<bean id="transactionManager"
 class="org.springframework.orm.hibernate3.HibernateTransactionManager">
<property name="sessionFactory">
<ref bean="sessionFactory" />
</property>
</bean>
```

```
<!-- 配置事务的传播特性 -->
<tx:advice id="txAdvice" transaction-manager="transactionManager">
<tx:attributes>
<tx:method name="add*" propagation="REQUIRED" />
<tx:method name="save*" propagation="REQUIRED" />
<tx:method name="del*" propagation="REQUIRED" />
<tx:method name="update*" propagation="REQUIRED" />
<tx:method name="modify*" propagation="REQUIRED" />
<tx:method name="*" read-only="true" />
</tx:attributes>
</tx:advice>
<!--哪些类的哪些方法参与事务-->
<aop:config>
<aop:pointcut id="allManagerMethod" expression="execution(* com.hrl.dao.*.*(..))" />
<aop:advisor pointcut-ref="allManagerMethod" dvice-ref="txAdvice" />
</aop:config>
```

（1）根据关键字搜索文章的前台 index.jsp 代码如下：

```
<table width="480" border="0" align="center" cellpadding="0"
cellspacing="0">
<s:hidden name="article.articleTypeName"
id="articleTypeName"></s:hidden>
<tr>
<td width="378" height="35">
<table width="359" height="35" border="0" cellpadding="0"
cellspacing="0">
 <tr>
 <td align="center">
<input type="text" id="searchStr"
                 name="searchStr" style="width: 350px; height:
20px;" />
</td>
 </tr>
</table>
</td>
<td width="113">
<img src="images/so.GIF" width="109"
 height="35" style="cursor: hand;" onclick="doSearch()"
/>
</td>
</tr>
</table>
```

（2）单击“搜索答案”按钮执行的 JavaScript 方法如下：

```
/**
* 搜索文章
* @return
*/
function doSearch() {
var searchText = $.trim($('#searchStr').val());
if (!searchText) {
alert('请输入要搜索的内容');
return;
}
if (searchText.length > 255) {
alert('输入内容不能超过 255 个字符');
return;
}
$('#articleTypeName').val(activeId);
doSearchForm.submit();
}
```

（3）在 Action 中主要负责接收参数，然后调用 DAO 层的 doSearch()方法。ArticleAction 具体代码如下：

```
/ * *
 * 通过关键字搜索文章
 * @return
 */
```

```
public String doSearch() {
        if (searchStr != null) {
                searchStr = searchStr.trim();
        }
        String type = this.article == null ? null : this.article.getArticleTypeName();
        this.searchArticles = this.articleDao.doSearch(type, searchStr, this.getFirstResult(), this.getMaxResults());
        return "searchResult";
}
```

（4）DAO 层采用 QBC 方式进行查询，这种方式的优点是不用手动编写 HQL 语句，也不用考虑一些 SQL 关键字的注入攻击（例如%、*、[]等特殊字符），只需要简单调用 Criteria 提供的简单方法即可。ArticleDaoImpl 具体代码如下：

```
/ * *
*  根据输入内容搜索符合条件的文章
*/
@SuppressWarnings("unchecked")
public List<Article> doSearch(String type, String str, String firstResult,String maxResults) {
        int first = new Integer(firstResult).intValue();
        int max = new Integer(maxResults).intValue();
        Criteria criteria = this.getCriteria(Article.class);
        if (type != null && !type.equals("")) {
                criteria.add(Restrictions.eq("articleTypeName", type));
        }
        criteria.add(Restrictions.or(Restrictions.like("title", str,MatchMode.ANYWHERE),
                  Restrictions.like("content", str,MatchMode.ANYWHERE)))
                 .addOrder(Order.desc("lastUpdateTime")).setFirstResult(first).setMaxResults(max);
        List<Article> list = criteria.list();
        return list;
}
```

心法领悟 600：Hibernate 中的 Criteria 对象。

Criteria 对象可以添加多个表达式，如本实例中添加表单时有根据关键字在任意位置进行匹配、按照文章的最后更新时间进行倒序排序和分页表达式。Criteria 使得开发者可以写更少的代码去完成更多的功能。